MIKROSZÁMÍTÓGÉP MAGAZIN

1986
október
Ára 30 Ft

Szellemgrafika
Helyi hálózatok
Játékprogramok
Barátunk a számítógép

Bencir

VI. SZERVEZÉSTECHNIKAI ESZKÖZÖK ÉS AZOK ALKALMAZÁSA SZAKKIÁLLÍTÁS ÉS KONFERENCIA

* Budapest Sportcsarnok *
1986. október 3—8.

1986.
OKTÓBER 3—8-ig

BUDAPEST SPORTCSARNOK NÉPSTADION ÚT 2.

MIKROSZÁMÍTÓGÉP MAGAZIN

4. ÉVFOLYAM
9. SZÁM
1986. OKTÓBER

A NEUMANN JÁNOS SZÁMÍTÓGÉP-TUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

E számunkat szerkesztették:
Bakos Tamás (programozástechnika)
Broczkó Péter (hírek)
Kovács Győző (levelezés)
Lindner László (sakkprogramozás)
Petróczy Judit (könyvek)
Simonyi Endre (klub)
Varga András (iskola-számítógép)

Pálhalmi Vali
PR menedzser

Felelős szerkesztő:
Könyves Tóth Pál
Szerkesztőség:
1027 Budapest II., Fő u. 68.
Telefon: 154-250

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat
Felelős kiadó:
dr. Petrus György igazgató
Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivataloknál
és a Posta Hírlapelőfizetési és Lapellátási Irodáján
(1900 Budapest V., József nádor tér 1.)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.
Egy szám ára 30,— Ft
Előfizetési díj:
fél évre 180,— Ft
egy évre 360,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, pf. 149.
és a Magyar Média
1932 Budapest, pf. 279. 86—253

Szikra Lapnyomda
Budapest (86-3951)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Címképünk:

ORGTECHNIK '86

Tartalom

Kettesben . . . 2
Assemblerek, cross-assemblerek 11
A minőségügy — közügy 20
A számítástechnika nem ámítástechnika 21
Olvastunk . . . 24
Helyi hálózatok 26
Ki ad magyarázatot? 33
Barátunk, a számítógép 38
Adok-veszek-cserélek 44

ISKOLA — SZÁMÍTÓGÉP

Az ismeretlen C16 3
Hangkeltés HT-gépen FORTH-ban 4

DIÁKROVAT

Programönkritika is . . . 5
Szövegszerkesztő és elemző 7
Szellemgrafika I. 8
Villogó grafika 9
Zeneprogram 10

PROGRAMOZÁSTECHNIKA

BASIC RAM felosztás 13
BASIC és gépi kód 15
Z80 programozási gyakorlatok 3. 16

µPROGRAMOK

PTA—4000. Egyváltozós függvények ábrázolása és megoldása 18
Tártartalom a képernyőn 19

JÁTÉKPROGRAMOK

A játékprogramozás technikája 22
Black Jack 23

µKLUB

PTA—4000. ROM leírás I. 29
A ZX81 bővítése 30
Botkormány 31
Príma ötlet 33

AZ OLVASÓ ÍRJA 34

SAKKPROGRAMOZÁS

Bitek és figurák 42

KÖNYVEK 45

HÍREK, ÉRDEKESSÉGEK 48

Kettesben...

„Tanítani az ifjakat komoly szóval körülkeríteni a telket, és mert a világ bús tengere forr, *felajzani a szilaj szenvedelmet, szólni a kor* göztorlatairól, *felverni az* alvó nemzeti lelket." *(Lator László: A hallgató Berzsenyi)*

A figyelmes olvasó bizonyára észrevette, hogy a µM-nak már nemcsak egy tulajdonosa van, a Neumann János Számítógéptudományi Társaság, hanem kettő, ugyanis 1986. július 1-jétől, ahogy a lap címoldalán is megjelent, a KISZ KB a lap társtulajdonosa.

A lapot az Ifjúsági Lapkiadó Vállalat gondozza, a szerkesztőségben a KISZ KB munkatársai is dolgozni fognak. Megalakul a lap szerkesztőbizottsága is; a bizottság elnökét és két tagját az NJSZT, három tagját a KISZ KB és egy tagját a Tudományszervezési és Informatikai Intézet delegálja. A szerkesztőbizottság feladata lesz a lap irányvonalának a meghatározása, tanácsadás, a szerkesztőség támogatása a felmerült problémák megoldásában.

Az előzetes megbeszéléseken egyetértettünk abban, hogy a lap alapkoncepciója, struktúrája nem változik, de szeretnénk a lapot még tartalmasabbá tenni, esetleg még a megjelenését is korszerűsíteni. Az elmúlt hónapokban jó kapcsolat alakult ki az Ötlettel, amelyet tovább szeretnénk erősíteni, hiszen egy magazin jellegű havilap és egy hetilap együttműködése rendkívül sok előnyt jelenthet mindkét sajtótermék számára.

A KISZ-nek és az NJSZT-nek számos olyan terve van, amelyet — a lap szerkesztésében való együttműködés mellett — közösen szeretnénk megvalósítani. Ezek közül a legfontosabbak például a µklub mozgalom kiterjesztése, a számítógépes amatőrizmus támogatása és fejlesztése, a távoktatási rendszer megvalósítása, amelyeket — összefogással — rövidebb idő alatt és valószínűleg gazdaságosabban lehet megszervezni, mint külön-külön, arról nem is beszélve, hogy ha külön úton járunk, akarva-akaratlanul még nehézségeket is támaszthatnánk egymásnak.

Szeretnénk a — kissé elaludt — µKönyvtár sorozatot is folytatni, már előkészületben van a következő, a Műszaki Könyvkiadó gondozásában megjelenő kötet, H. H. Goldstine informatikatörténeti munkája, amelynek címe: „A számítógép Pascaltól von Neumannig". Nagyon szoros együttműködésünk van a Magyar Televízióval (Mi és a számítógép, a TV BASIC és az újabb oktatófilmek), de elkezdődik a munka a Magyar Rádióval is. A legközelebbi tervünk egy közös kéthetenkénti magazinműsor, amelyet ugyancsak most készítünk elő. A Tudományszervezési és Informatikai Intézettel és így a magyar oktatásüggyel a szoros kapcsolatunkat nem csak a TII munkatársának szerkesztőbizottsági tagsága jelzi, de a távoktatási rendszert is úgy szeretnénk megvalósítani, hogy az a magyar köz- és felsőoktatás része legyen.

A Neumann Társaság természetes állandó szövetségesei voltak a főhatóságok, szeretnénk, ha a KISZ KB-vel való együttműködésben is számíthatnánk a támogatásukra, és így magasabb szintre emelhetnénk kapcsolatunkat az OMFB-vel, az IpM-mel, a KSH-val és az MM-mel.

Meggyőződésünk, hogy igazán nagy és komoly eredményt — ma amikor az anyagi körülmények egyre rosszabbodnak — csak összefogással, a meglevő szellemi energiáknak és anyagi erőknek a koncentrálásával lehet elérni.

Ehhez valamennyiüknek a legjobbakat kívánom.

KOVÁCS GYŐZŐ

ORGTECHNIK '86

A hazai szervezéstechnika nagy találkozója a kétévente megrendezésre kerülő ORGTECHNIK kiállítás, valamint a vele párhuzamosan folyó tanácskozás. Itt egymáshoz mérhetik magukat a szervezéssel és szervezéstechnikai segédeszközök gyártásával foglalkozó hazai nagy- és kisvállalkozások, de arra is lehetőség nyílik, hogy megismerhessék: vajon hol állnak e területen a külországbeli cégek. Az idei kiállítást, amely a hatodik a hasonló rendezvények sorában, október 3—8. között a Budapest Sportcsarnok arénájában és galériáján tartják, s a kiállításhoz kapcsolt kongresszusra pedig október 6—8. között kerül sor. Az idei rendezvényről beszélgettünk dr. Pompéry Bélával, aki a Szervezési és Vezetési Tudományos Társaság Szervezési Szakosztálya, valamint a Számítógépalkalmazási Munkabizottság részéről fogja össze a szervezés feladatait.

— Milyen újdonsággal, érdekességgel találkozhatunk az idei rendezvényen?

— A kiállítás célja elsődlegesen a szervezési, számítástechnikai módszerek bemutatása. A szervezési módszer azonban nem olyan látványos dolog, amelyiket érdeklődést keltően ki lehetne állítani. Így a külső szemlélő óhatatlanul egy eszközkiállításnak, a szervezéstechnikai segédeszközök kiállításának lesz a látogatója. A technikának ez esetben a szervezést kell szolgálnia, s amint az előzetes jelentkezésekből látható, a kiállítók messzemenően igyekeznek így összeválogatni a bemutatójukat. Természetes, hogy a számítástechnika egyre inkább bevonul a mindennapi életbe. Így van ez a szervezéstudományban is, ahol a korábban fejben vagy papíron végzett adatelemzési, rendezési, statisztikai rutinmunkát elvégzi a számítógép. De az ipari szervezés részét képezik például az ipari robotok is . . .

— Ezek szerint nem csak számítógépeket, adatfeldolgozó berendezéseket láthat a publikum?

— A kiállításon részt vesz 5 nagy szocialista egyesülés a baráti országokból. Hazánk 110 nagyvállalatával és sok kisvállalkozással képviseltetni magát. Önálló standdal jelentkezik az Alkotó Ifjúság Egyesülés, a Videoton, a Terta, SZKI, SZÜV, Novotrade, Csepel, SZÁMALK, a Magyar Posta, valamint az 5G kisszövetkezet, hogy a várhatóan legérdekesebb hazai bemutatókat említsem, a teljesség igénye nélkül szemezgetve az előzetes listából.

— Tőkés országokból is várnak kiállítókat?

— Még nem zárultak le a jelentkezések véglegesen, így korai lenne e tárgyban pontos adatokat mondani. Mintegy 20 cég jelezte részvételét. Már biztosnak látszik a REX-Rotary másológép, a Robinco fototechnikai, a DUIT cég mikrofilmtechnikai berendezéseinek kiállítása. S hogy a számí-

Az ismeretlen C16

KERNAL rutinok 2.

A beviteli/kiviteli KERNAL rutinok egy összefüggő csoportjának leírásával folytatom a C16 ismertetését. Ezek közül a BSOUT az előző részben is szerepelt, most részletezem az ott leírtakat.

Az alábbiak között öt olyan rutin van, amely átvitelbit beállításával jelzi, hogy a végrehajtás hibátlan volt-e. Ha nem volt hiba, a C-bitet törli, ellenkező esetben a C-bitet 1-re állítja, és az A regiszterbe teszi a hiba kódját. Ajánlatos a hívó programrészbe a következő szerkezetű ellenőrző részt beépíteni:

```
JSR rutin
BCC TOVABB
JMP ($0300)
TOVABB ...
```

A rutinok

SETLFS $FFBA 65 466

A fájl-paramétereket helyezi el az erre kijelölt rendszerváltozókban. Ezek általában megegyeznek a BASIC OPEN utasításának első három paraméterével. Hívása előtt a paramétereket a regiszterekbe kell tölteni, az alábbiak szerint:

A — a logikaifájl-szám,
X — a fizikai eszközszám, ez megegyezik a BASIC-ben is használatossal,
Y — másodlagos cím (megnyitási mód, ill. csatornaszám).

SETNAM $FFBD 65 469

A fájlnévre vonatkozó információkat helyezi el a kijelölt rendszerváltozóba. Hívása előtt az A regiszterbe a fájlnév karaktereinek a számát (a név hosszát) kell tölteni, név nélküli fájl esetén 0-t. Az X/Y regiszterpár a fájlnév kezdőcímét kell hogy tartalmazza úgy, hogy a cím nagyobb helyértékű bájtja kerüljön Y-ba.

OPEN $FFCO 65 472

Megnyitja azt a fájlt, melynek paramétereit előzőleg a SETLFS és a SETNAM rutinokkal az erre kijelölt rendszerváltozókba küldtük. Hiba esetén a bevezetőben leírtak szerint működik.

CLOSE $FFC3 65 475

Lezárja azt a fájlt, melynek logikai számát híváskor az A regiszter tartalmazza.

CHKIN $FFC6 65 478

Beviteli csatornát készít elő olvasásra. Hívása előtt az X regiszterbe annak a korábban OPEN-nel megnyitott fájlnak a logikai számát kell elhelyezni, melyből a következő olvasások történnek a BASIN rutinnal. A hibakezelés itt is a bevezetőben leírtak szerint történik.

CHKOUT $FFC9 65 481

Kiviteli csatornát készít elő írásra. Az adatáramlás irányától eltekintve megegyezik a CHKIN rutinnal.

CLRCHN $FFCC 65 484

Lezárja a CHKIN és CHKOUT rutinokkal megnyitott csatornákat, és visszaállítja az alaphelyzetet, azaz a továbbiakban az aktuális beviteli eszköz a billentyűzet, a kiviteli a képernyő lesz.

BASIN $FFCF 65 487

Beolvas egy karaktert az aktuális beviteli csatornáról az A regiszterbe. Alaphelyzetben (CHKIN használata nélkül, ill. CLRCHN után) a billentyűzetről olvas, a képernyőszerkesztő segítségével.

BSOUT $FFD2 65 490

Kiküld egy karaktert az A regiszterből az aktuális kiviteli csatornára. Alaphelyzetben (CHKOUT használata nélkül, ill. CLRCHN után) a képernyő kurzorpozíciójába ír, és — néhány kivételtől eltekintve — a kurzort egy hellyel jobbra lépteti.

LOAD $FFD5 65 493

Programfájl betöltő/ellenőrző rutin. Hívása előtt a fájl adatait a SETLFS és SETNAM rutinok segítségével be kell állítani. Ha hívásakor az A regiszter 0-t tartalmaz, betölti a programot a háttértárolóról, ha 1-et, akkor összehasonlítja a tárban lévő programot a háttértárolón lévővel. Ha a SETLFS hívásakor a megnyitási mód kódja 0 volt, a betöltési kezdőcímet az X/Y regiszterpárba kell a LOAD hívása előtt beírni, 1-es megnyitási mód esetén a betöltési címet a fájlból veszi.

A cím regiszterbe írásakor az Y-ba kell a nagyobb helyértékű bájtot tölteni, X-be az alacsony helyértékűt. Hibátlan végrehajtás után a regiszterpár a végcím + 1 értéket fogja tartalmazni.

A hibakezelésre itt is a bevezetőben leírtak érvényesek.

SAVE $FFD8 65 496

A programfájlt a háttértárolóra viszi. Használata előtt a fájl jellemzőit a SETNAM és SETLFS rutinokkal be kell állítani. Ezután a program kezdőcímét egy nulla-lapos bájtpáron kell elhelyezni, az A regiszterbe ennek a bájtpárnak a címét kell tölteni. Az X/Y regiszterpárba az első olyan bájt címét kell bevinni, amely már nem tartozik a programhoz (végcím + 1). A címet úgy kell a regiszterekbe tenni, hogy az Y regiszterben legyen a cím magasabb helyértékű bájtja.

A kezdőcím elhelyezésére még később visszatérek, most annyit ajánlok, hogy C16 esetén $D8 ... $D9 (216 ... 217), a többi géptípus esetén $FB ... $FC (251 ... 252) című helyek alkalmasak erre a célra. Ne feledkezzünk meg arról, hogy itt is az alacsonyabb helyértékű bájtot kell előre írni!

A hibakezelés itt is a bevezetőben leírtak szerint történik.

CLALL $FFE7 65 511

A CLRCHN hívásával lezárja a megnyitott csatornákat, majd lezárja a megnyitott fájlokat, és visszaállítja az alaphelyzetet.

BARNA LÁSZLÓ

---

tástechnika se maradjon ki: itt van például az INTERMEC, amelyik vonalkódtechnikai berendezéseket és a COMBIRING, amely mikroszámítógépeket mutat be. S várhatóan sok egyéb érdekességgel is találkozhat a látogató, aki október 3—8. között, naponta 10—18 óráig felkeresi a Budapest Sportcsarnokban lévő bemutatóstandokat. Itt mindenkit, laikust, szakembert vagy éppen diákot szívesen látnak a kiállítók. De mindenkit arra kérek: ne csak az elektronikát nézze! Az csak eszköz egy probléma megoldására.

— Meghívottak részére a kiállítás vége felé tudományos konferenciát is rendeznek, idén már másodízben. Mi ennek a programja?

— Kettős céllal rendezzük: először is azért, hogy aki kiállít, az elmagyarázhassa előadásában, milyen módszereket alkalmaz a gyakorlatban. Aki viszont nem állít ki, de alkalmazza a szervezéstechnikát, az is ismertesse tapasztalatait. Mind a kongreszszus, mind a kiállítás alkalmat nyújt arra, hogy szolgáltató és felhasználó megtalálhassa egymást. Hazai feladat van elegendő, s vállalkozók is akadnak ennek megoldására. A feladatunk az, s ezért szervezzük ezeket a rendezvényeket a Szervezési Vezetési Tudományos Társasággal, a Neumann János Számítógéptudományi Társasággal és az Országos Vezetőképző Intézettel közösen, hogy nőjön a szervezés iránti igényesség, a partnerek egymásra találhassanak, mint egy szakvásáron, s nem utolsósorban a szakma egy helyen láthassa: ilyen a magyar szervezési kultúra. Nézzétek! Itt tartunk, s azt is láthatjuk, merre kellene tovább fejlődnünk.

KIS

Hangkeltés HT gépen FORTH-ban

Ha a FORTH rendszerben nincs beépített assembler szótárunk, akkor is írhatunk egyszerűbb gépi kódú primitíveket. Néha erre kifejezetten szükség van. Vegyünk egy példát. A FORTH kiválóan alkalmas különféle hardvereszközök vezérlésére. Ha azonban a BASIC-nek megfelelő INP és OUT utasítások nincsenek meg, akkor éppen a legfontosabb láncszem hiányzik. A fenti szókat kellene használnunk a HT iskolaszámítógépen a hang létrehozásához is. Bizonyára többen fordulnának a FORTH nyelvhez, ha könnyen elérhető lenne néhány ilyen praktikus dolog.

Először is létre kell hoznunk azokat a primitíveket, amelyek a portokat kezelik. A gépi utasítások felépítését assembler hiányában magunknak kell megszerveznünk. Segítségképpen az OUTC és INPC primitíveket közöljük. A kódokat legegyszerűbben a C, és a, FORTH szókkal rakhatjuk le. Tudnunk kell, hogy a CREATE szó egy primitív fejrészt hoz létre, és a SMUDGE szó érvényesíti azt. Próbaképpen hozzuk létre a DUP FORTH szót újra. Ez elég rövid ahhoz, hogy megérje kipróbálni. A szó egyébként a rendszerben így van megvalósítva *(1. lista)*.

A sor lezárásakor a DUP IS NOT UNIQUE üzenet jelenik meg, mert a DUP szó már megvan a szótárban. Ez azonban most csak azt jelenti, hogy az új DUP van érvényben. A primitívek létrehozásakor néhány dologra ügyelnünk kell. A FORTH rendszer a működésekor egymáshoz láncolt primitíveket hajt végre. Néhány különleges esettől eltekintve ezek mindegyike egy meghatározott rutinra ugrik, amely gondoskodik a következő primitív megkereséséről. Ez a rutin az implementációtól függő helyen kezdődik, fizikailag általában a primitívek előtt helyezik el. Az általunk használt rendszerben a 4CBAH, 4CBBH vagy 4CBCH címen kezdődik, attól függően, hogy akarunk-e a veremre eredményt elhelyezni. Ez a rutin feltételezi, hogy a Z80 processzor BC regiszterpárját azonos állapotban hagytuk. Amennyiben mégis használjuk, a tartalmát meg kell őriznünk és vissza kell töltenünk.

Ha a DUP szó újradefiniálása sikerült, bátran nekifoghatunk egy komolyabb feladatnak. Hosszabb primitívek létrehozásához jobb módszert ajánlunk. A *2. listában* bemutatjuk az assembler listát, a *3. listában* a FORTH megvalósítást.

A LERAK szót azért hoztuk létre, hogy hosszabb primitíveket kényelmesebben lehessen beírni. A szó egy a verem tetején hagyott szám alapján a veremről sorba az aktuális HERE címre rakja az adatokat. Előnye, hogy a gépi utasításokat a rendes sorrendben lehet megadni. Ezek ugyan a vermen fordított sorrendben jelennének meg, a LERAK szó azonban a verem aljáról indulva bányássza ki őket. Vigyázni kell a használatával, mert az SP! szó újra inicializálja a vermet.

Az OUTC és INPC szókban a C végződés arra utal, hogy a primitívben a port címét a C regiszter adja meg.

Ha a létrehozott szókat nemcsak hangkeltésre akarjuk használni, akkor az INIC, KIMEN és BEHOZ szók alkalmasak a perifériával való kommunikációra.

A CVEK egy definiáló szó. Példánkban a TENG szót hoztuk létre vele. A szó definiálásakor a szó neve után rendre lerakja a szó előtt megadott egybájtos számokat, a megadás sorrendjében. Itt a sorozat legelején azonban azt találjuk majd, hogy hány eleme van a sorozatnak. A definiált szó végrehajtásakor egy-egy számot vár a vermen, és az ennyiedik bájtot fogja ráhelyezni. Közben ellenőrzést is végez, hogy létezik-e egyáltalán annyiadik szám. CVEK segítségével tehát olyan szókat hozhatunk létre, amelyek tárolják a portra kivinni kívánt információt.

A *4. listán* egy dallam programozását mutatjuk meg. A H1, H2, H3 vektorváltozóba tettük a hangkódokat, az S1, S2, S3 vektorváltozóba pedig a hosszúságot. A HANG szók adják a dallam egyes részeit. Mellékeljük a dallam BASIC változatát is *(5. lista)*, hogy legyen módja az olvasónak összehasonlítani. Néhány érdekességre felhívjuk a figyelmet.

BASIC-ben a hangzás hosszúságát egy 30-szoros ciklus adja; itt ugyanezt egy 7 hosszúságú belső és 80-szoros külső ciklus adja. Külön kellett definiálni egy S szót a hangváltások közé, mert az azonos magasságú hangok összefolytak.

NYIRATI LÁSZLÓ

```
( TASK )
      DUP SZO
           1. E1         POP HL
           2. E5         PUSH HL
           3. C3BB4C     JP 4CBBH
  UGYANEZ FORTH-BAN

HEX CREATE DUP SMUDGE E1 C, E5 C, C3 C, 4CBB , DECIMAL

VEGREHAJTVA:
   3 DUP . . (HL)
A VALASZ:
 3 3 OK
```

1. lista

```
OUTC
   1. E1           POP HL; ADAT
   2. 7D           LD A,L
   3. E1           POP HL; PERIFERIA
   4. C5           PUSH BC; BC MEGORZESE
   5. E5           PUSH HL
   6. C1           POP BC; C FELTOLTVE
   7. ED79         OUT (C),A; ADAT KI
   8. C1           POP BC; EREDETI BC VISSZA
   9. C3BC4C       JP 4CBCH; UGRAS A VISSZATERESRE

INPC
   1. E1           POP HL ; PERIFERIA
   2. C5           PUSH BC ; BC MENTESE
   3. E5           PUSH HL
   4. C1           POP BC ; PERIFERIA CIME A C-BE
   5. ED78         IN A,(C); ADAT AZ A-BAN
   6. C1           POP BC ;  REGI TARTALOM VISSZA
   7. 6F           LD L,A
   8. 2600         LD H,0
   9. C3BB4C       JP 4CBBH ; UGRAS VISSZATERESRE
                              ADAT ATADASSAL
```

2. lista

```
( TASK )
: LERAK 2 * SP@ DUP ROT + DO I @ DUP . C, -2 +LOOP SP! ;
HEX CREATE OUTC SMUDGE E1 7D E1 C5 E5 C1 ED 79 C1 C3 BC
4C 11 LERAK
CREATE INPC SMUDGE E1 C5 E5 C1 ED 78 C1 6F 26 0 C3 BB 4C D
LERAK DECIMAL : INIC 31 7 OUTC 30 128 OUTC ;
: KIMEN 31 15 OUTC 30 SWAP OUTC ;
: BEHOZ 31 14 OUTC 31 INPC ;
: CVEK <BUILDS DUP 1+ C, LERAK DOES> 2R DUP R C@ 1+ < R>
SWAP IF 1+ + C@ ELSE ." HIBA" QUIT THEN ;
0 0 0 0 0 0 31 199 16 16 16 255 40 16 0 0 16 CVEK TENG
: HANG 16 0 DO 31 I OUTC 30 I TENG OUTC LOOP ;
: CSEND 16 0 DO 31 I OUTC 30 0 OUTC LOOP ;



( LERAK --> HERE CIMRE N BYTE-T LERAK )
( OUTC, INPC PRIMITIVAK )
( INIC --> AZ A CSATORNA BEMENETKENT A B KIMENETKENT MUK. )
( KIMEN/BEHOZ --> INIC UTAN EGY KI/BE ADATATVITELI LEHETOSEG )
( CVEK N BYTE-OS VEKTOR LETREHOZASA )
( b1 b2 b3.....bn n CVEK nev --> nev-VEL EGY SZO )
( VEGREHAJTASKOR  sorszam nev --> bs )
( TENG A TENGERZUGAS ADATAI ) ( HANG --> TENGERZUGAS )
( CSEND --> A KIMENETI EGYSEGEKET 0-VAL TOLTI FEL )
```

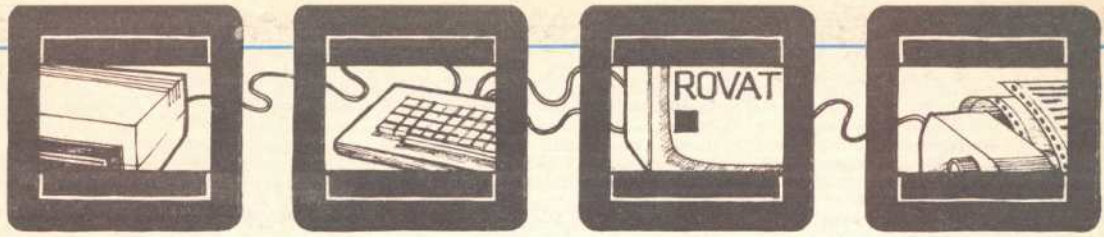
3. lista

```
( AUTOMOB )
152 152 152 135 135 135 120 113 101 152 10 CVEK H1
161 152 135 203 4 CVEK H2 135 161 152 3 CVEK H3
2 2 4 2 2 4 2 2 2 2 10 CVEK S1 2 2 2 2 4 CVEK S2
2 2 4 3 CVEK S3
: CIK 7 0 DO LOOP ; : S 31 0 OUTC 30 0 OUTC CIK ;
80 VARIABLE KES : SZUM KES @ * 0 DO CIK LOOP ;
: HANG1 10 0 DO S 31 0 OUTC 30 I H1 OUTC I S1 SZUM LOOP ;
: HANG2 4 0 DO S 31 0 OUTC 30 I H2 OUTC I S2 SZUM LOOP ;
: HANG3 3 0 DO S 31 0 OUTC 30 I H3 OUTC I S3 SZUM LOOP ;
: HANG0 31 7 OUTC 30 254 OUTC 31 8 OUTC 30 9 OUTC ;
: AUT HANG0 HANG1 HANG2 HANG1 HANG3 CSEND ;
```

4. lista

5. lista

```
10 DATA 152,2,152,2,152,4,135,2,135,2,135,4,120,2,113,2,101,2,152,2,161,2,152,2,
135,2,203,2
20 DATA 152,2,152,2,152,4,135,2,135,2,135,4,120,2,113,2,101,2,152,2,135,2,161,2,
152,4
30 OUT31,7:OUT30,254:OUT31,8:OUT30,9
40 FORL=1 TO 27
50 READ P,Q
60 OUT31,0:OUT30,P
70 FOR T=1 TO Q*30:NEXTT
80 OUT31,0:OUT30,0
90 NEXTL
```



PROGRAMÖNKRITIKA IS...

A µMagazin 1986. júniusi számában megjelent Ékezetes ábécé C64-re című cikkünkben a BASIC kezdőcím eltolását végző parancssor hibás. Helyesen:

POKE44,16:POKE43,65:POKE4096
+64,0:NEW

Itt a második parancs 43-as címe helyett 45 jelent meg. A cikk többi része helyes, kivéve az 1. program 45Ø-es sorának szövegében lévő helyesírási hibát és a javasolt elhelyezés SHIFT oszlopában szereplő, teljesen felesleges ö betűt.

A cikk megjelenése után elég sűrűn kerestek telefonon, egyrészt a hiba (43 helyett 45), másrészt általában a BASIC kezdőterület eltolása, illetve a használható nyomtató típusa miatt. Vegyük sorra a kérdéseket!

Az első esetben nem mentegetőzöm, én írtam el gépeléskor! Ezért kötelességemnek tartom egy hibátlan, általánosan használható BASIC kezdő- és végcímeltolást végző parancssor leírását programban is felhasználható módon, kiegészítve még néhány fontos technikai megoldással.

Az ékezetes ábécét használva eddig VC 1525, MPS 801 és EPSON GX típusú nyomtatókon nyomtattunk, ezeken működött. Kaptam olyan telefont is, hogy MPS 802-n az ékezetes betűk helyett a megfelelő grafikus karaktereket nyomtatja ki a program. Ehhez nem tudok hozzászólni, eddig nem nyomtattam 802-esen.

A BASIC terület kezdete a 43, 44 címeken lévő érték LO—HI sorrendben. Alaphelyzetben LO=1 HI=8. Azaz a BASIC kezdőcím a

KE=PEEK(43)+256*PEEK(44)

alapján számítható ki. A BASIC által használt legmagasabb címet az 55, 56-os címeken találjuk szintén LO—HI sorrendben. Alaphelyzetben LO=0, HI=160. A BASIC végcím tehát a

VE=PEEK(55)+256*PEEK(56)

összefüggés alapján megállapítható. A BASIC kezdő-, illetve végcímeltolás több okból is szükségessé válhat (új karakterkészlet, gépi rutin, szellemek helye, bittérképes kép tárolása stb.). A BASIC kezdete előtti bájt tartalma kötelezően nulla! A kezdő- és végcímek megváltozását az interperter csak akkor veszi tudomásul, ha kiadjuk a NEW vagy CLR utasítást! A feladat általában az, hogy a BASIC kezdő- vagy végcímét akarjuk átállítani valamilyen CIM-re. A betöltendő LO—HI értékek a CIM=LO+256*HI összefüggés alapján számolhatók, ezért a kezdő terület a

POKE43,CIM—256*INT(CIM/256)
POKE44,INT(CIM/256)
POKE CIM—1,0:NEW

a végcím a

POKE55,CIM—256*INT(CIM/256)
POKE56,INT(CIM/256):NEW

1. program

```
0 REM ***** INDITO *****
1 :
2 X=X+1:IFX=1THENGOSUB40
3 ONXGOTO5,6,10
4 PRINT"■HIBA":STOP
5 LOAD"KEP",8,1
6 LOAD"KEP BE",8,1
7 :
10 PRINT"■■LOAD"CHR$(34)"BETU"CHR$(34);
11 PRINT",8,1"
12 PRINT"■■■LOAD"CHR$(34)"MENU";
13 PRINTCHR$(34)",8"
14 PRINT"■■■RUN"
15 :
20 FORI=0TO3:READA:POKE631+I,A:NEXT
21 POKE198,4
22 DATA19,13,13,13
23 :
30 POKE43,1:POKE44,16:POKE4096,0
31 POKE55,0:POKE56,120:NEW
32 :
40 POKE53280,5:POKE53281,5
41 PRINT"■■■■■■■■■■■■■■■■■■■■"
42 PRINT"■■■■■■■■■■■■■■■■■■OLGOZOM !"
43 PRINT"■■■■■■■■■■■■■■■■■B. 1 PERC"
44 RETURN
```

2. program

```
0 REM ***** MENU *****
1 :
2 POKE53280,5:POKE53281,5:PRINT"■"
3 POKE53272,(PEEK(53272)AND240)OR2
4 REM SZINEK, KARAKTERKESZLET ALLITAS
5 RESTORE
6 :
10 PRINT"■■■■■■■■■■■■■■■■■■■■MENU"
11 PRINT"■■■■■■■■■■■■■■■■■■■■";
12 PRINT"■■■■■■■■■■■■■■■■■■■■";:REM C= + P
13 PRINT"■■■■■■■■  PROGRAM VEGE"
14 PRINT"■■■■■■■1  NAGY1 PROGRAM"
15 PRINT"■■■■■■■2  NAGY2 PROGRAM"
16 PRINT"■■■■■■■3  NAGY3 PROGRAM"
17 PRINT"■■■■■■■4  NAGY4 PROGRAM"
18 :
20 POKE198,0:A$=""
21 GETA$:IFA$=""THEN21
22 X=ASC(A$)-48
23 IFX<0ORX>4THEN20
24 :
30 ONXGOTO50,60,70,80
31 :
32 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■";
33 PRINT"ENNYI VOLT":SYS64738
34 :
40 PRINT"■■■■■■■■■■■■■■■■■■" POKE198,3
41 PRINT"■■■■■■■■■■■■■■■■■■OLGOZOM !"
42 PRINT"■■■■■■■■■■■■■■■■■■B. 1 PERC"
43 :
44 FORI=0TO2:READA:POKE631+I,A:NEXT
45 DATA19,13,13
46 LO=PEEK(45):HI=PEEK(46):IFLO<2THEN48
47 POKE43,LO-2:POKE44,HI:RETURN
48 POKE43,254+LO:POKE44,HI-1:RETURN
49 :
50 GOSUB40:PRINT"■■LOAD"CHR$(34)"NAGY1",
51 PRINTCHR$(34)",8":PRINT"■■■RUN"
52 CLR:STOP
53 :
60 GOSUB40:PRINT"■■LOAD"CHR$(34)"NAGY2",
61 PRINTCHR$(34)",8":PRINT"■■■RUN"
62 CLR:STOP
63 :
70 GOSUB40:PRINT"■■LOAD"CHR$(34)"NAGY3",
71 PRINTCHR$(34)",8":PRINT"■■■RUN"
72 CLR:STOP
73 :
80 GOSUB40:PRINT"■■LOAD"CHR$(34)"NAGY4",
81 PRINTCHR$(34)",8":PRINT"■■■RUN"
82 CLR:STOP
83 :
94 END
```

3. program

```
0 REM ***** NAGYX *****
1 :
2 POKE53280,5:POKE53281,5:PRINT"■"
3 :
4 REM EZEN KIVUL EGYEB INICIALIZALAS
5 :
10 PRINT"■■■■■■■■■■■■■■■■■■■■MENU"
11 PRINT"■■■■■■■■■■■■■■■■■■■■";
12 PRINT"■■■■■■■■■■■■■■■■■■■■";:REM C= + P
13 PRINT"■■■■■■■■  VISSZA A FO MENURE"
14 PRINT"■■■■■■■1  FUNKCIO 1"
15 PRINT"■■■■■■■2  FUNKCIO 2"
16 PRINT"■■■■■■■■"
17 PRINT"■■■■■■■■"
18 PRINT"■■■■■■■9  FUNKCIO 9"
19 :
20 POKE198,0:A$=""
21 GETA$:IFA$=""THEN20
22 X=ASC(A$)-48
23 IFX<0ORX>9THEN20
24 :
30 ONXGOTO51,52,53,54,55,56,57,58,59
31 :
32 POKE53280,5:POKE53281,5:PRINT"■";
33 RESTORE
34 PRINT"POKE43,1:POKE44,16:CLR"
35 PRINT"■■RUN":POKE198,3
36 FORI=0TO2:READA:POKE631+I,A:NEXT
37 DATA19,13,13
38 REM A 37-ES AZ ELSO DATA SOR !!!!!
39 NEW:STOP
40 :
41 :
51 REM TENNIVALO FUNKCIO 1 VALASZTASKOR
52 REM TENNIVALO FUNKCIO 2 VALASZTASKOR
53 REM TENNIVALO FUNKCIO 3 VALASZTASKOR
54 REM TENNIVALO FUNKCIO 4 VALASZTASKOR
55 REM TENNIVALO FUNKCIO 5 VALASZTASKOR
56 REM TENNIVALO FUNKCIO 6 VALASZTASKOR
57 REM TENNIVALO FUNKCIO 7 VALASZTASKOR
58 REM TENNIVALO FUNKCIO 8 VALASZTASKOR
59 REM TENNIVALO FUNKCIO 9 VALASZTASKOR
60 END
```

parancsokkal beállítható. A NEW helyett CLR is használható.

Egy-egy ilyen átállítás végrehajtása a program minden egyes felhasználása előtt fáradságos, ugyanakkor az is elképzelhető, sőt valószínű, hogy a program a működéséhez az esetleges eltolások mellett igényli más programok jelenlétét is. Ezek parancsokkal történő mindenkori betöltése megnehezíti mind a programozó, mind a felhasználó dolgát.

A fenti problémák automatikus megoldására, a felhasználó helyzetének megkönnyítésére valók a betöltő- (LOADER, BOOT) programok. Megoldásuk többféle lehet; a leggyakrabban előforduló, általánosan is használható típusokat ismertetem. A cél kettős: egyrészt tájékoztassuk a felhasználót az eseményekről, hogy nyugton maradjon; másrészt végezzük el a szükséges betöltéseket, átállításokat. Ez utóbbiakhoz a felhasználónak semmi köze, csak zavarná. Előle úgy tüntetjük el a zajló eseményeket, hogy a neki szánt világos, megnyugtató, a képernyő alján elhelyezett kiírás után egyeztetjük a tinta és a papír színét, így a betöltésekhez, átállításokhoz szükséges kiírások a képernyő tetején a felhasználó előtt láthatatlanok maradnak.

A feladat megoldásához tudnunk kell még, hogy az utoljára lenyomott billentyűk ASCII kódjait tároló billentyűzetpuffer a 631—640 címeken van. Azt, hogy pillanatnyilag hány értékes (max. 10) lenyomott billentyű van a pufferben, a 198-as cím tárolja. Ismert még, hogy a LOAD utasítás végrehajtása után a program futása annak első sorától folytatódik, ezért mindenképpen gondoskodnunk kell a betöltőprogram helyes működéséről. Erre szolgál az ON X GOTO utasítás, melynek szemantikáját az esetleges programhibák kiküszöbölésére is felhasználhatjuk. Tudjuk ezenkívül, hogy a LOAD"név",8,1 utasítással betöltött programok, fájlok a mentési helyükre kerülnek vissza, míg a LOAD"név",8 utasítás mindig a BASIC terület aktuális elejétől tölti be a programot.

Az előzőek mellett találkozhatunk még azzal a problémával is, hogy egyszerre két vagy több BASIC programot szeretnénk elhelyezni a rendelkezésünkre álló BASIC területen, illetve olyan programrendszert írtunk, melynek programjai nem férnek el egyszerre a tárban. Ekkor feltétlenül szükséges egy olyan rövid menüprogram megírása is, mely biztosítja az egyes fő BASIC programok tetszőleges sorrendben történő behívását, indítását. Célunk az, hogy a felhasználó feladata az általunk írt programrendszer futtatásakor mindössze annyi legyen, hogy behívja és elindítja a betöltőprogramot. A többi automatikusan menjen anélkül, hogy a felhasználó arról tudomást szerezne, beavatkozna. A technikai részletek pedig végképp nem tartoznak rá.

Tegyük fel, hogy írtunk egy olyan programrendszert, mely a NAGY1, NAGY2, NAGY3, NAGY4 nevű BASIC programok mellett tartalmaz egy BETU nevű ékezetes ábécé programot (helye 2 k—4 k között), egy KEP nevű, KOALA PAINT programmal készített képet (helye 30 k—40 k között), valamint ennek előhívását és a szükséges oda-vissza kapcsolásokat végző KEP BE nevű gépi kódú programot (helye a kazettapuffer). KEP és KEP BE valahol a NAGYX programokban kerülnek felhasználásra. NAGYX programjaink olyan hosszúak, hogy kettő együtt már nem fér el a tárban, viszont mindegyik mellett van még hely egy rövid BASIC program számára. A NAGYX programok négy különböző feladatot oldanak meg, a felhasználók ezeket egy-egy futtatás alkalmával tetszőleges sorrendben, tetszőleges sokszor hívhatják.

Feladatunk: írjuk meg az INDITO nevű betöltőprogramot, mely a kezdő- és végcím átállítása mellett betölti a helyükre a BETU, KEP, KEP BE programokat, valamint behívja és elindítja az ugyancsak most megírandó MENU programot, mely a felhasználó választása alapján behívja és elindítja a NAGYX programok valamelyikét. A felhasználó NAGYX programok mindegyikéből visszatérhet a MENU programra. Ezt a visszatérést végző részt is bemutatom, a programrészletet közlöm.

Először vegyük szemügyre az INDITO program szerkezetét! Első feladata, hogy tájékoztassa a felhasználót az eseményekről. Ezután helyükre töltheti a KEP, KEP BE programokat. BETU betöltésével viszont már baj van, hisz éppen INDITO helyére kerül! Betöltése előtt feltétlenül át kell állítani a BASIC terület kezdetét, aminek végrehajtása a NEW miatt megöli INDITO-t! KÉP is csak azért tölthető be a végcím átállítása előtt, mert INDITO rövid és nem használ szöveg típusú változókat. Ha a kezdőcímet átállítottuk, akkor átállíthatjuk a végcímet is.

Hogyan töltjük hát be BETU és MENU programokat és indítjuk el az utóbbit? Úgy, hogy az átállítások előtt a képernyő tetejétől kezdődően a megfelelő sorokba PRINT-tel kiíratjuk a betöltéseket és az indítást végző parancsokat, majd a billentyűpuffert feltöltjük a megfelelő vezérlő karakterekkel (HOME, RETURN), ugyanis ezek a programfutás végén a pufferből automatikusan a képernyőre kerülnek, biztosítva a parancsok végrehajtását. A felhasználó előtt mindez rejtve marad, ha a papír és a tinta színe megegyezik. Próba alatt azonban jobb, ha látjuk, mi történik (lásd *1. program*). Még annyit: CHR$(34) az idézőjel, 19 a HOME, 13 a RETURN kódja.

MENU lesz az a program, amely a mindenkori behívott NAGYX mellett még elfér a tárban. Lényegében — a felhasználó szempontjából — egy menüből áll, ahol a program vége választás mellett a mindenkori NAGYX behívása és indítása választható. Emellett mindössze annyit tud még, hogy INDITO-hoz hasonlóan a BASIC kezdőcím további átállításával eltünteti, de nem semmisíti meg önmagát, mielőtt behívná és indítaná NAGYX-et, valamint átállítja a karakterkészletet az ékezetes betűkre. Közben természetesen értesíti a felhasználót a programbetöltés tényéről. Az eljárás logikája ugyanaz, mint INDITO-é az átkapcsolások végrehajtásakor, illetve BETU és MENU betöltésekor, indításakor.

A különbség mindössze annyi, hogy ott (INDITO írásakor) előre tudtuk az új kezdőcím értékét, míg most csak akkor tudnánk, ha MENU már kész lenne. A probléma megoldható úgy is, hogy elkészítem MENU-t, a végcím helyét kipontozom, majd a kész program után PRINTPEEK-kel lekérdezem a címet és a pontok helyére beírom. De az így elkészített program a legkisebb változtatás után már rosszul működik. Sokkal biztosabb, ha a feladatot általánosan oldjuk meg. Így az esetleges későbbi változtatások mellett is működik, használható. Tudjuk már, hogy a BASIC kezdőcím előtti bájt értéke kötelezően nulla. Ez a feltétel most automatikusan teljesül, hiszen az utolsó BASIC sor vége ügyis 0, amit a program végét jelző két darab 0 követ.

A BASIC program végét a 45, 46-os címeken tárolja a gép LO—HI sorrendben. MENU tulajdonképpen NAGYX betöltése és futtatása előtt lekérdezi e címekről saját végét, majd a BASIC kezdetét ennél két bájttal előbbre állítja be, törölve a program végét jelző két nullát. Az utolsó sor végét jelző nulla megmarad. Csak ezután tölti be és indítja a választott NAGYX-et. Tulajdonképpen ezt a technikát alkalmazzuk programok összefűzésekor is, azzal a különbséggel, hogy ott az utolsó program betöltése után visszaállítjuk a BASIC első program előtti kezdetét, és az így egybefűzött programokat egy programként mentjük lemezre. Visszatérve MENU-re, a felsoroltak mellett még azt is tudja, hogy ha a lemezen nincs meg véletlenül a választott NAGYX, akkor a lemezegység piros lámpája villog ugyan, de a program nem áll le, a képernyőn újra a menü lesz látható. A lámpa villogása egy, a programba épített, inicializáló utasítással kivédhető. Részletesebben lásd a *2. programot*.

Mi a leírt technika előnye? Ha a felhasználó egy-egy futtatás alkalmával sűrűn hívja a különböző NAGYX-eket, akkor nem kell várnia MENU újbóli betöltésére, az átkapcsolás egy pillanat műve.

Így már érthető, hogy NAGYX-ek menüjében miért legyen visszatérési lehetőség a MENU programra. A visszatérési, programvége-lehetőség általában az aktuális menü 0 választási pontja, hiszen így nehezen téveszthető, lásd a billentyűzetet. NAGYX programoknak csak a menüjét és a MENU programra való visszatérést végző részét írtam meg *(3. program)*.

Mindhárom programban zöld a keret és a papír színe, fehér a felhasználónak szóló üzenet és a menü szövege, zöld viszont a tinta színe az átállító, betöltő, futtató parancsok kiírásakor. Ismét hangsúlyozom, hogy a próbák alatt ajánlatos a tinta színét végig fehéren tartani! Futás közben a képernyő figyelmes tanulmányozása választ adhat a még felmerülő kérdésekre. INDITO csak akkor futtatható, ha valóban létezik BETU, KEP, KEP BE.

ÉNEKES FERENC

Kedves Olvasóink!

Nem rendelésre készült kéziratokat nem őrzünk meg és nem küldünk vissza. Levelek, cikkek stb. közlése esetén szerkesztőségünk fenntartja a jogot az írások rövidítésére!

Szövegszerkesztő és elemző

A program 48 k-s géphez készült, ugyanis egymástól függetlenül 40 képernyőt kezelhetünk (karakteres felbontás), de fut 16 k-s gépen is, amennyiben csak egy képernyőre van szükségünk s többet nem akarunk kezelni.

A program indítás után a nevét írja ki, gombnyomás után hozzáfoghatunk a szerkesztendő szöveg beírásához. Használhatóak a kis- és nagybetűk, számok, valamint a SYMBOL SHIFT-tel elérhető jelek, például írásjelek. A karaktereket keret alakú, villogó kurzor követi.

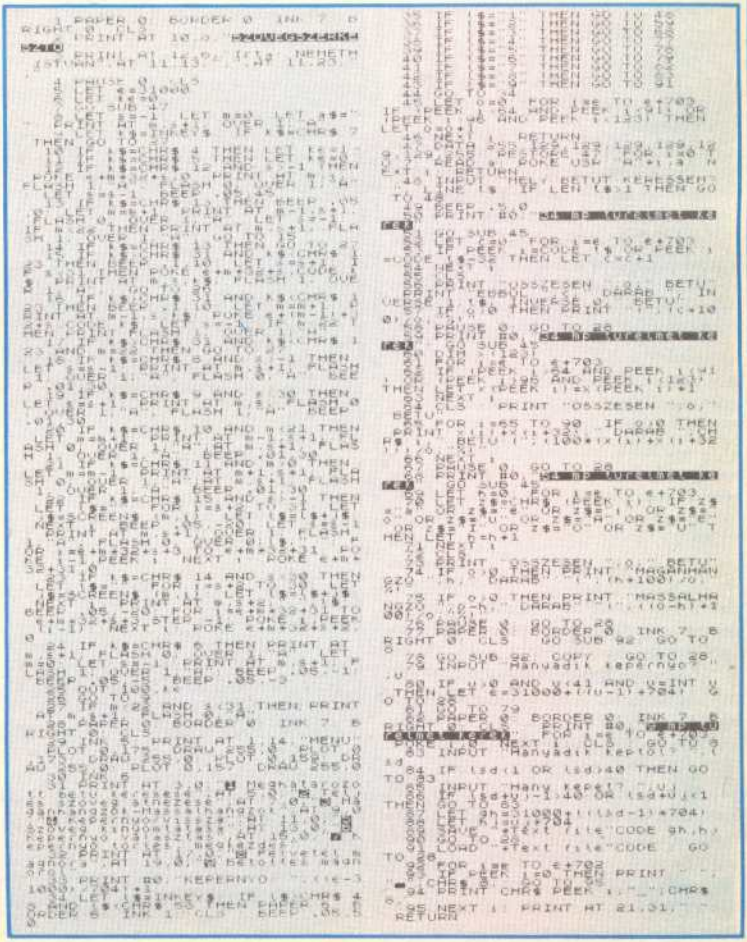
Ha a sor végére érünk, a következő betű automatikusan új sorba kerül. Az ENTER gomb lenyomásával is kezdhetünk új sort. Ha mellétütünk, a DELETE gomb (CAPS SHIFT +0) leütésével törölhetünk egyet vissza. Ezenkívül a kurzort bárhova mozgathatjuk a képernyőn a nyílgombok segítségével (CAPS SHIFT +5, +6, +7, +8). Rámehetünk vele egy karakterre, s az általunk kívánt betűre cserélhetjük az adott billentyű leütésével. A CAPS LOCK segítségével (CAPS SHIFT +2) a kurzor a bal felső sarokba ugrik (Commodore-on: CURSOR HOME). Ha egy hosszabb sort gépeltünk be, de az elején kihagytunk egy szót, akkor a kérdéses helyre megyünk, s a SYMBOL SHIFT, CAPS SHIFT együttes lenyomásával a kurzortól jobbra elhelyezkedő karaktereket jobbra tolhatjuk, helyet szúrtva a kimaradt szónak. Így tehát probléma nélkül szúrhatunk be szavakat. Ha viszont sok SPACE-t tettünk egy szó elé, vagy a sor túlságosan jobbra csúszott, akkor a GRAPHICS gombbal (CAPS SHIFT +9) balra húzhatjuk a kurzortól jobbra elhelyezkedő részeket.

Egyébként a programban minden betű leütését hangjelzés követi. Van egy különleges funkció is, amelyet a TURE VIDEO gombbal (CAPS SHIFT +3) kapcsolhatunk be, és az INV VIDEO-val (CAPS SHIFT +4) kapcsolhatunk ki. Ha ezt a funkciót használjuk, akkor a keret nem fekete, hanem kék lesz, s egy gomb lenyomásáig vált vissza feketére. Ha kiírta a karaktert, ismét kék lesz, s várja a következő betűt. Mindez a másodperc tört része alatt történik, de így csak a billentyűzetre kell figyelni, a keret változásaiból érzékelhetjük a megfelelő tempót. A program a beírt szöveget a 31 000. memóriacímtől tárolja karakterenként. Szerkesztés közben is közvetlenül ide írja be a friss adatokat.

Ha a szerkesztést befejeztük, az EDIT gombbal (CAPS SHIFT +1) szállhatunk ki, s tekinthetjük meg a MENÜ-t. Itt az utasítások arra a képre vonatkoznak, amelynek sorszáma alul is megjelenik. Az 1-es gomb lenyomására a gép kikeresi a kívánt betűt, kiírja, mennyi van belőle összesen, s ez az egésznek hány százaléka. A 2-es gombbal a teljes szöveg elemzését kérhetjük. Várakozás után kiíródik minden betű darabszáma és százalékos aránya. A 3-as gombbal a magánhangzók és mássalhangzók arányát tudhatjuk meg. A programmal tehát nyelvészeti elemzéseket is végezhetünk. A 4-es gombbal visszatérhetünk az aktuális képhez, folytathatjuk a szerkesztést. Az 5-ös gombbal printerre kinyomtatható a szöveg. A 6-os gomb lenyomásával megválasztjuk, hányadik képernyőn akarunk dolgozni, 1—40-ig. A 7-es gombbal egy képernyő tartalmát törölhetjük. A 8-as gombbal felvehetjük tetszőleges számú képernyő tartalmát kazettára. Először az első felveendő képet kell megadni, majd azt, hogy összesen hányat. Például, ha az első négy képet kívánjuk rögzíteni, akkor a beadandó számok: 1;4. A 9-es gombbal betölthetjük a felvett képeket.

A program beírásánál ügyeljünk arra, hogy a 12—24-es soroknál az idézőjelben lévő A betűk grafikusak.

NÉMETH ISTVÁN
ELTE Ságvári Endre Gyakorló Iskola



SZELLEMGRAFIKA I.

A szellem (sprite) grafika, és a vele kapcsolatos műveletek a C64 speciális lehetőségei közé tartoznak. Problémamentesen egyszerre maximum 8 szellem jeleníthető meg a képernyőn, annak bármilyen üzemmódjában. Ezt a számot csak a megszakítások menetének biztos ismeretében, annak felhasználásával léphetjük át, kihasználva az optikai csalódást.

Egy szellem BASIC-ből történő felhasználásához az alábbi műveletek szükségesek:

- alakjának megtervezése, az azt előállító 63 bájt létrehozása, betöltése;
- a helymutató beállítása;
- a normál vagy többszínű üzemmód kiválasztása;
- a szín meghatározása, prioritások beállítása;
- nagyítás beállítása;
- a kezdőhelyzet beállítása, bekapcsolás;
- az előbbiek folyamatos változtatása, esetleg a megszakítások kezelése, ütközésvizsgálat.

Vegyük sorra az egyes műveleteket! A szellem alakjának megtervezésére több módszer ismert, a „kockás" papírtól a SPRITE EDITOR programokig. Egy szellem alakját 3×21 bájt határozza meg, ahol minden bekapcsolt pontnak 1-es érték felel meg, azaz egy szellem 24×21 képpontból áll. Ezt a 63 bájtot kell sorfolytonosan bevinni a memória megfelelő helyére. Mivel a szellemeket a VIC—II chip kezeli, ezért azok adatait az általa használt 16 k-s szelet valamely 64-gyel osztható címétől kezdődően helyezzük el.

Az egyszerre használható 8 szellem — sorszámaik 0—7 — alakját meghatározó adatok kezdetére egy-egy mutató mutat, melyek értékei rendre a 2040-2047 címeken találhatók. A VIC—II a szellem adatainak kezdőcímét e mutatók és a szeletsorszám (0—3) alapján a cím = 16384 × szeletsorszám + 64 × mutató összefüggés alapján határozza meg. Természetesen nem használható minden 64 bájtos lap! A 11, 13, 14, 15-ös sorszámú helyek minden további nélkül használhatók. A 32—63 sorszámúak a BASIC kezdőcím 4 k-ra, a 128—255 helyek a 16 k-ra történő eltolás mellett érhetők el.

A szellem — a bekapcsolt pontok — színe a 16 szín bármelyike lehet. A kikapcsolt pontokban azt látjuk, ami a szellem mögött egyébként a képernyőn van. A szellemek színregiszterei az 53287—53294 címeken találhatók. A többszínű üzemmód kiválasztása az 53276 címre írt érték n. bitjének magasra állítását jelenti. Ekkor az n. sorszámú szellem többszínű üzemmódban jelenik meg. A többszínű üzemmód azonos a karaktergrafikai megfelelőjével, a szellem soronkénti 24 bitje helyett 12 bitpár jelenik meg, melyek színei:

00 — átlátszó,
01 — többszín-regiszter 1,
10 — színregiszter,
11 — többszín-regiszter 2.

A többszín-regiszter 1 az 53285, a többszín-regiszter 2 az 53286 címeken található.

Két szellem találkozásakor a kisebb sorszámú a nagyobb sorszámú előtt jelenik meg, azaz a kisebb sorszámú eltakarja a nagyobbat. Külön intézkedni csak a szellem és a képernyő kitöltött részeinek egymáshoz viszonyított helyzetére vonatkozóan kell. Ha azt akarjuk, hogy az n. szellem eltakarja a hátteret, akkor az 53275-ös cím n. bitjét állítsuk ALACSONYRA! Ez a kezdő érték is.

A szellemek megjeleníthetők normál méretben, vagy X, Y, illetve mindkét irányban kétszeresre nagyítva. A nagyítási pont a szellem bal felső sarka. A vízszintes, X irányú nagyítás az 53277-es, a függőleges, Y irányú nagyítás az 53271-es címre írt értékkel szabályozható. Ha a címre írt érték n. bitje magas, akkor az n. szellem az adott irányban kétszeresre nagyítva jelenik meg. A felbontás nem lesz finomabb.

A szellem helyének megadásakor mindig a bal felső sarkának koordinátáit adjuk

```
100 REM *********************************
101 REM *      EZ ITT A FOPROGRAM       *
102 REM *********************************
103 PRINT"□" GOSUB126
104
105 FORX=0TO197 GOSUB207 GOSUB207
106 FORL=0TO20 NEXTL
107 GOSUB223 NEXTX
108
109 FORC=0TO110 GOSUB213 GOSUB213
110 GOSUB219 NEXTX
111
112 FORC=0TO190 GOSUB223 GOSUB219
113 FORL=0TO20 NEXTL,C
114
115 FORC=0TO85 GOSUB223
116 GOSUB213 GOSUB219
117 GOSUB213 GOSUB219 NEXTC
118
119 GOSUB219 FORL=0TO20 NEXTL
120 IFK<>-1THENGOTO119
121 RESTORE GOSUB126 GOTO105
122
123 REM *********************************
124 REM *     VIC  INICIALIZALAS        *
125 REM *********************************
126 F0=2040 F1=F0+8 F2=53248 F3=2368
127 FORA=0TO7 POKEF1+32*9+A,0 NEXTA
128 FORA=2TO7 READB POKEF1+A,B NEXTA
129 FORA=0TO3 READF(A) NEXTA
130 FORA=0TO2 POKEF1+A,F(0) NEXTA
131 IFPEEK(F3)=2THEN138
132
133 FORA=0TO383 READB A%=A%+B
134 POKEF3+A,B NEXTA
135 IFA%=20173THEN138
136 PRINT"*** DATA HIBA ***" END
137
138 POKEF0,37 POKEF2+28,1
139 POKEF2+39,0 POKEF2+37,10
140 POKEF2+38,1 POKEF2+1,101
141 POKEF2+21,1 POKEF2,0
142 POKEF2+16,0
143
144 PRINT"[illegible]",
145 PRINT"[illegible]",
146 PRINT"[illegible]"
147 POKEF2+32,12 POKEF2+33,11
148 POKEF2+34,8 POKEF2+35,7
149 POKEF2+24,19 POKEF2+22,216
150 RETURN
151
152 DATA255,192, 49, 12,255
153 DATA 90,150,165,105
154
155 DATA  2,128,  0, 10,160,  0, 10, 80
156 DATA  0,  9,208,  0,  9, 84,  0,  9
157 DATA 80,  0,  1, 64,  0,  3,192,  0
158 DATA 15,240,  0, 13,240,  0, 13,240
159 DATA  0, 13,240,  0, 15,112,  0, 10
160 DATA 96,  0,  2,128,  0,  2,128,  0
161 DATA  2,128,  0,  2,128,  0,  3, 64
162 DATA  0,  2,128,  0,  2,160,  0,  0
163 DATA  2,128,  0, 10,160,  0, 10, 80
164 DATA  0,  9,208,  0,  9, 84,  0,  9
165 DATA 80,  0,  1, 64,  0, 15,  0,  0
166 DATA 63,192,  0, 55,192,  0, 55,192
167 DATA  0, 31,208,  0, 55,192,  0, 41
168 DATA120,  0, 10,  0,  0, 10,128,  0
169 DATA 10,160,  0, 10,128,  0, 12,  0
170 DATA  0, 10,  0,  0, 10,128,  0,  0
171 DATA  0,  0,  0,  2,128,  0, 10,160
172 DATA  0, 10, 80,  0,  9,208,  0,  9
173 DATA 84,  0,  9, 80,  0, 12,  0,  0
174 DATA 63,  0,  0, 55,192,  0, 21,208
175 DATA  0,127,192,  0, 21,192,  0, 72
176 DATA120,  0, 10,128,  0, 10,160,  0
177 DATA 10, 40,  0, 42,224,  0, 52,144
178 DATA  0, 40, 40,  0, 10,  0,  0,  0
179 DATA  0,  0,  0,  2,128,  0, 10,160
180 DATA  0, 10, 80,  0,  9,208,  0,  9
181 DATA 84,  0,  9, 80,  0, 13,  0,  0
182 DATA 63,  0,  0, 55,192,  0, 55,212
183 DATA  0, 21,192,  0, 55,192,  0, 41
184 DATA128,  0, 10,128,  0, 42,160,  0
185 DATA232,160,  0,144,160,  0,163, 64
186 DATA  0, 24,128,  0,  0,160,  0,  0
187 DATA  2,128,  0, 10,160,  0, 10, 80
188 DATA  0,  9,208,  0,  9, 84,  0,  9
189 DATA 80,  0,  1, 64,  0, 15,  0,  0
190 DATA 63,192,  0, 55,192,  0, 55,208
191 DATA  0, 31,192,  0, 55,192,  0, 41
192 DATA128,  0,202,128,  0,154,160,  0
193 DATA154,160,  0,130,128,  0,131, 64
194 DATA  0,  2,128,  0,  0,160,  0,  0
195 DATA  2,128,  0, 10,160,  0, 10, 80
196 DATA  0,  9,208,  0,  9, 84,  0,  9
197 DATA 80,  0,  1, 64,  0,  3,192,  0
198 DATA 15,240,  0, 13,240,  0, 13,240
199 DATA  0, 13,240,  0, 13,240,  0,  9
200 DATA224,  0,  2,160,  0,  2,168,  0
201 DATA  2,160,  0, 10,128,  0, 13, 64
202 DATA  0, 10,128,  0,  2,160,  0,  0
203
204 REM *********************************
205 REM * SZUBRUTINOK A  FOPROGRAMHOZ   *
206 REM *********************************
207 IFPEEK(F2)<>255THEN211
208 P=PEEK(F2+16) POKEF2,0
209 IFP=0THENPOKEF2+16,1 RETURN
210 POKEF2+16,0 RETURN
211 POKEF2,PEEK(F2)+1 RETURN
212
213 IFPEEK(F2)<>0THEN217
214 P=PEEK(F2+16) POKEF2,255
215 IFP=0THENPOKEF2+16,1 RETURN
216 POKEF2+16,0 RETURN
217 POKEF2,PEEK(F2)-1 RETURN
218
219 IFK<>9THENK=-3
220 FORA=0TO2 POKEF1+A,F(K) NEXTA
221 K=K+1 RETURN
222
223 IFPEEK(F0)=42THENPOKEF0,37 RETURN
224 POKEF0,PEEK(F0)+1 RETURN
```

meg. Elhelyezésük az úgynevezett szellemtérben történik, mely egy 512×256-os tóruszfelület, lényegesen nagyobb, mint a 320×200-as képernyő. A koordináták értékei a szokásos módon, 0-val kezdődnek. A tóruszfelület azt jelenti, hogy a szellemtér alja-teteje, bal és jobb oldala összeér. A szellemtér képernyőhöz viszonyított helyzete a következő: a szellemtér (24;50) koordinátájú pontja a képernyő bal felső sarka a (0;0) koordinátákkal. Mivel a szellem helyének X koordinátája 0—511 közé eshet, tárolására nem elegendő egy bájt, egy további, 256 értékű bit is szükséges; ezt a bitet nevezzük MSB-nek. Ha az n. szellem MSB bitje magas, akkor az X koordinátát meghatározó bájt értékéhez 256 adódik hozzá. A szellemek MSB bitjét meghatározó bájt címe 53264. Ennek n. bitje az n. szellem MSB-jét kapcsolja be-ki. Az 53248, 53249, . . ., 53262, 53263 címek rendre az n. szellem helyzetének X, Y koordinátáit határozzák meg.

Az n. szellem bekapcsolása, megjelenítése az 53269 címen levő érték n. bitjének magasra állításával érhető el. Hogy a szellem a képernyőn látszik-e vagy sem, azt természetesen a pillanatnyi helyzetét megadó X, Y koordinátapár dönti el.

Ahhoz, hogy a megadott álló szellem mozogjon, legalább a helyzetét meghatározó X, Y koordinátapár változtatása szükséges. Egy figura mozgatásához ezenkívül legtöbbször a mozgást jellemző fázisokat változtatjuk úgy, hogy azokat egy-egy szellem jeleníti meg, és ezeket kapcsoljuk ki-be. A VIC—II biztosítja azt a lehetőséget is, hogy megvizsgáljuk két szellem, illetve egy szellem és a háttér ütközésének megtörténtét is. Ütközésen mindig a bekapcsolt pontok ütközése értendő. Két szellem természetesen akkor is ütközhet, ha nincsenek a látható képernyőn. Az 53278 című regiszter két szellem, az 53279 a szellem-háttérütközéseket regisztrálja. Ütközéskor az elsőnél mindig kettő, az utóbbinál egy bit értéke lesz magas. Olvasásukkal (PEEK) az ütközés megtörténtét ellenőrizhetjük. Vigyázat, olvasáskor mindkét regiszter törlődik, ezért értéküket a további vizsgálathoz kötelező elmenteni! Többszínű üzemmódban a 01 bitkombinációjú pontok ütközés szempontjából áttetszőnek számítanak. Az ütközések vizsgálata a program elágaztatását szolgálja.

Végül az X. cím N-edik bitjének magasra állítását a POKE X,PEEK(X) OR 2↑N, alacsonyra állítását a POKE X,PEEK(X) AND (255-2↑N) utasítás végzi el. A szellemek BASIC-ből történő felhasználása elég körülményes, sokszor a gyorsaság sem megfelelő. Lehetőségeik valódi kihasználását csak gépi kódban érhetjük el.

A program címe Menő Manó. Begépelése előtt toljuk el a BASIC terület kezdetét 4 k-ra az ismert módon. A program egy szellemet jelenít meg, majd mozgat hat fázisban többszínű karaktergrafikai támogatással. Felépítése:

103	Kiindulási értékek beállítása helyben részletezve.
105—107	A szellem mozgatása: két rácspont jobbra, időzítés, fázisváltás.
109—110	A szellem kétszer balra, és @ karakter átírása.
112—113	A szellem helyben mozog fázisváltással.
115—117	Fázisváltás, szellem és karakterkép mozgatása balra.
119—121	A karakterkép fázisváltásának kivárása, kezdet beállítása, vissza az elejére.
126	F0: 0. szellem, F1: 2 k eleje, F2: VIC—II eleje, F3: szellembetöltési cím a 37-es blokkba.
127	A szellem adatainak nullázása
128	@ alsó öt bájtja lesz.
129	@ felső három bájtjának négy fázisa.
130	Az első fázis betöltése.
131	Ha a szellem betöltve, akkor nem tölt újra.
133—136	DATA sorok betöltése, ellenőrző összeg.
138	0. szellem a 37-es blokkban, szellem többszínű üzemmód be.
139—140	0. szellem színe fekete, többszínregiszterek világos piros és fehér, Y koordináta.
141—142	X koordináta, MSB.
144—146	Egy sor már új @.
147—149	Karakter többszín színei, többszínű üzemmód be, 0 szeletre átkapcsolás.
152—153	@ új adatai.
155—202	A szellem hat fázisának adatai, fázisonként nyolc sor.
207—211	A szellem jobbra mozgatása, MSB kapcsolása.
213—217	A szellem balra mozgatása, MSB kapcsolása.
219—221	Karakterkép fázisváltása.
223—224	A szellem fázisváltása.

A szellem egy közismert játék főszereplője.

FÖLDI ENDRE—ÉNEKES FERENC

DIÁK HT-1080z ROVAT

Villogó grafika

A program a grafikus karakterek inverzét állítja elő, ezáltal villogtatja a képernyőt. A programot megadjuk BASIC-ben DATA sorokba építve és assemblerben is. A BASIC programnak van egy egyszerű része, amellyel kipróbálhatjuk a szubrutint.

BARANYI ZSOLT

A BASIC program

```
  5 REM BETÖLTÉS
 10 CLS
 20 FOR C=28672 TO 28708
 30 READ A : POKE C,A : NEXT C
 40 REM DEMO
 50 X=RND(127) : Y=RND(47)
 60 SET(X,Y)
 70 IF INKEY$="E" THEN 80 ELSE 50
 80 POKE 16526,0:POKE 16527,112
 90 FOR T=0 TO 50 : REM VILLANÁSOK SZÁMA
100 Z=USR(0)
110 FOR C=0 TO 120:NEXT C
120 REM SZÜNET
130 NEXT T
140 REM A GÉPI KÓDÚ RÉSZ
150 DATA 33,0,60,6,0,14,4,126,254,32
160 DATA 40,16,254,128,56,4,254,192
170 DATA 56,12,119,35,16,239,13,32,236
180 DATA 201,62,191,24,244,47,198,64,24
190 DATA 239
```

A program ASSEMBLER-ben

```
      LD HL,3C00H; A KÉPERNYŐ ELSŐ PONTJA
      LD B,0
      LD C,4
CIKL:LD A,(HL)
      CP 32; SPACE?
      JR Z,FEHER ; HA IGEN, UGORJ FEHERRE
      CP 128 ; KARAKTER?
      JR C,KI ; HA IGEN, UGORJ KIRE
      CP 192 ; GRAF. JEL?
      JR C,INVERZ ; HA IGEN, UGORJ INVERZBE
KI:LD (HL),A
      INC HL
      DJNZ CIKL
      DEC C
      JR NZ,CIKL
      RET
FEHER:LD A,191
      JR KI
INVERZ:CPL
      ADD A,64 ; A 6. BIT NULLÁZÁSA
      JR KI
```

DIÁK primo ROVAT

Zeneprogram

Ezzel a programmal dallamokat csalhatunk ki számítógépünkből. A dallam öt szakaszból áll. Az adatok bekérésekor az első szám a hosszúságot, a második pedig a magasságot jelenti. Máris játszhatunk.

MUTSCHLER PÉTER

```
0 REM ZENEGENERALAS PRIMORA
5 REM MUTSCHLER PETER 1986
10 CLS
20 PRINT CHR$(2)
30 PRINT"    ZENEGENERALAS"
40 PRINT CHR$(1)
50 PRINT:PRINT:PRINT
60 PRINT"EZZEL A PROGRAMMAL DALLAMOKAT CSALHATSZ KI GEPEDBOL."
65 PRINT"(HA LEHET 1500-NAL NAGYOBB SZAMOT NE IRJ BE)"
70 PRINT:PRINT"ELSO SZAM A HOSSZUSAG,A MASODIK A MAGASSAG"
72 REM ************************
74 REM *   ADATOK BEKERESE    *
76 REM ************************
80 PRINT"A DALLAM ELSO SZAKASZA"
90 INPUT C,B
100 PRINT:PRINT"A MASODIK SZAKASZ"
110 INPUT C,D
120 PRINT:PRINT"A DALLAM HARMADIK SZAKASZA"
130 INPUT E,F
140 PRINT:PRINT"A DALLAM NEGYEDIK SZAKASZA"
160 INPUT H,J
170 PRINT"BEFEJEZO SZAKASZ"
180 INPUT K,L
190 REM ****************************
192 REM *  A ZENE MEGSZOLALTATASA  *
194 REM ****************************
200 FOR A=0 TO 20
210 BEEP INT(A/0),B
220 NEXT A
230 BEEP C,D
240 BEEP E,F
250 FOR G=0 TO22
260 BEEP H,J
270 NEXT G
280 BEEP K,L
290 PRINT:PRINT
300 PRINT"AKARSZ MEG EGY MENETET? (I/N)"
310 INPUT M$
320 IF M$="I" THEN 10
330 IF M$="N" THEN 340
340 PRINT"HAT HA NEM HAT NEM "
350 PRINT CHR$(2):PRINT"SZIA":PRINT CHR$(1)
360 END
```

Az előző részben összeállított kapcsolatábra segítségével leírt rendszer működésére két példát mutatunk be: az első egy direktíva feldolgozásához, a második pedig a makróhívás felismeréséhez és a kifejtés indításához szolgál illusztrációként.

A példákban megadjuk a feldolgozandó forrásnyelvi sort, a rendszer állapotát és a feldolgozáshoz szükséges táblák és veremmemóriák tartalmát a feldolgozási ciklus kezdetekor, majd megvizsgáljuk a feldolgozással kapcsolatosan végrehajtandó feladatokat, s operátorról operátorra haladva végigkövetjük a feldolgozási ciklus folyamatát. Az elágazást megvalósító operátoroknál kijelöljük, hogy a T (igaz) vagy az F (hamis) ágon haladunk-e tovább.

1. példa

Legyen a rendszer az A1 állapotban:

PASS=1
DEF=Ø
IF1=x (közömbös)
IF2=Ø
MIND=Ø
LOC=1ØØ H (H-val hexadecimális formában megadott számot jelölünk)

A feldolgozás alatt lévő, SPB2-n tárolt forrásprogramsor legyen:

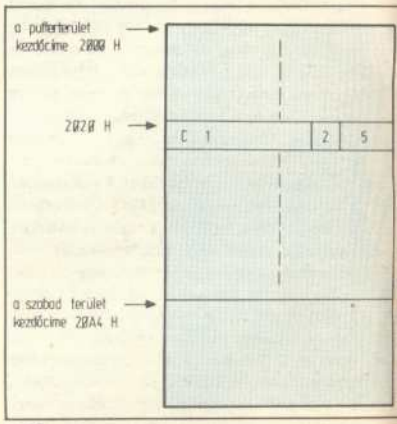
CIM1 DS C1+3

A végrehajtandó feladatok:

— a címkemezőben szereplő „CIM1" relatív szimbólum elhelyezése a modulcimketáblában;

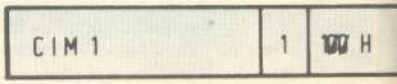
— a műveleti kód mezőben szereplő direktíva kiértékelése (LOC növelése), ehhez azonban előbb

— meg kell határozni az operandusmező értékét: (a szükségképpen predefinit



1. ábra

2. ábra



Formatervezett, megbízható működésű botkormánygyártmányaink között biztosan megtalálja az igényeinek megfelelőt! Az alaptípusú készülék termelői ára csupán 369 Ft.

Kérje katalógusunkat, amit megcímzett válaszboríték ellenében díjtalanul megküldünk!

INTER GM.

Kaposvár, Rákóczi tér 1. 7400

ASM–250 típusú szünetmentes áramforrás

Hálózatkimaradás esetén megszakítás nélkül min. 30 perc időtartamig biztosítja a tápfeszültséget. Névleges teljesítmény: 250 VA.

Érdeklődni lehet: ERFI – Erőáramú Gyártmány- és Rendszerfejlesztő Vállalat, Vállalkozási iroda 1027 Bp. Medve u. 25–29. Telefon: 359-740, 354-140. Telex: 22 50 62 erfi h.

ERFI

Assemblerek, cross-assemblerek

8. Példák a rendszer működésére

szimbólum) C1 értékét hárommal kell növelni;

— a sor megőrzése a második menet számára.

Tegyük fel továbbá, hogy a modulcímketábla pufferterületének tartalma az *1. ábra* szerinti, és a „CIM1" szimbólumot még nem definiáltuk. A táblából kitűnik, hogy a „C1" szimbólum korábban EQU direktívával (2-es címkeflag) 5-ös értéket kapott, azaz programunkban létezik az aktuális sor előtt valahol egy például

C1 EQU 5

alakú forrássor.

Az egymás után aktivizált operátorok a forrásprogramsornak az SPB2 területre való átvitele után a kapcsolatábrát követve az alábbiak.

Az operátor típusa		Az operátor funkciója
5	—	Megtörténik a mezőkre bontás: megjegyzésmező: üres, címkemező: CIM1, műveletikód-mező: DS, operandusmező: C1+3.
2	—	Elindul közben a következő forrásprogramsor olvasása, s ezzel párhuzamosan folytatódik az aktuális sor feldolgozása.
6	—	A T ág aktivizálódik, mert a megjegyzésmező üres.
6	—	IF2=Ø, tehát a T ág érvényes.
6	—	DEF=Ø, a T ágon haladunk tovább.
7	—	Mivel MIND=Ø, az operátor aktivizálása után közvetlenül be is fejezi a működését.
6	—	A címkemező 1. karaktere nem üres, tehát: F.
6	—	MIND=Ø, tehát: T.
8	—	Korábban nem definiáltuk a címkét, tehát: F.
9	—	Az operátor elkészíti a címkebejegyzést (*2. ábra:* a címke relatív, értéke 1ØØ H), s elhelyezi a címketáblában a 2ØA4 H címre, a szabad terület kezdőcímének mutatóját az új bejegyzésnek megfelelően 2ØA4H+8=2ØAC H-ra módosítja.
12	—	A műveletikód-mezőben nem makrónév szerepel: F.
14	—	A műveletikód-mező tartalma szerint keres a teljes műveletikód-táblában, megtalálja a DS direktívának megfelelő bejegyzést, ami tartalmazza, hogy — direktíváról van szó; — kötelező operandusa van a direktívának; — és megadja a 19-es operátoron belül a feldolgozó szubrutin kezdőcímét; a T ágon haladunk tovább.
15	—	Direktíva szerepel a műveletikód-mezőben: T.
16	—	Az operátor először is megállapítja, hogy az operandusmező nem üres és benne egy összetett kifejezés szerepel; szétválasztja az egyszerű kifejezéseket: C1 és 3, és a kijelölt műveletet: +. Felismeri az első kifejezés szimbólum jellegét, és kutatni kezd a modulcímketáblában, ahol a 2Ø2Ø H címen megtalálja a „C1"-hez tartozó bejegyzést és megállapítja, hogy a szimbólum egy abszolút kifejezés és értéke 5. Elvégzi az összeadási műveletet, s a következő operátornak a 8-as operandusértéket és egy indikátorban annak jelzését adja át, hogy az operandus abszolút típusú.
19	—	Az operátor elvégzi a DS direktívának megfelelő műveletet: — beállítja a tárfoglalás nagyságát jelző változót 8-as értékre.
20	—	SPB2 tartalma (a forrássor) lemezre kerül.
11	—	A címketábla még nem telt be, tehát lemezre való kiírásra nincs szükség.
6	—	Még nem érkezett be az END direktíva, IEND=Ø: T.
21	—	Lezárja a sorfeldolgozási ciklust; LOC értékét a tárfoglalás nagyságát megadó változó értékével növeli, az új érték 1Ø8 H.

Ezután indulhat újra a következő sorfeldolgozási ciklus.

Nyolcrészes sorozatunk a mikroszámítógépek assemblereiről, cross-assemblereiről szól. Célja, hogy a cross-assemblerek példáján keresztül megismertesse az olvasót az assembler programok működésével. A bemutatáshoz a rendszermodellezési eszközöket használjuk fel, s így készítjük el az assemblerek működésének egy szabványosított algoritmusát, modelljét. E cikkünk a sorozat befejező része.

1. **(Cross-)assemblerek és a rendszermodellezés**
2. **Az Intel 8080 assembly nyelv**
3. **A (cross-)assembler program mint rendszer**
4. **A rendszer működése I. — A fordítás két menete**
5. **A rendszer működése II. — Táblák és addatterületek**
6. **Az operátorkészlet I.**
7. **Az operátorkészlet II. — A rendszer kapcsolatábrája**
8. **Példák a rendszer működésére**

2. példa

Legyen a rendszer az MK1 makrókifejtési állapotban, s a feldolgozás alatt álló forrásprogramsorban szerepeljen újabb (beskatulyázott) makróhívás. A rendszerváltozók kezdeti értéke:

PASS=1

DEF=Ø
IF1=x (közömbös)
IF2=Ø
MIND=1
LOC=15A H.

Az első szinten kifejtett makróban a feldolgozásban soron következő sor legyen
CIM2 MULTP OP1,OP2,X3
alakú. Tegyük fel, hogy a makródefiníciós könyvtárunkban már létezik egy MULTP nevű makró, amelynek formája a következő:

```
MULTP MACRO X1,X2,X3
.
.
.
utasítások
.
.
.
MEND
```

azaz három formális paraméterünk van: legyen ezek közül X1 és X2 két operandus címe, az eredményt pedig az X3 címre kell elhelyezni. A hívósorban már láthatóan az első két formális paramétert az OP1 és OP2 aktuális paraméterek váltották fel (legyenek ezek modulcímkék), a harmadik paraméter viszont még nincs behelyettesítve. Ez csak úgy lehetséges, ha X3 egyben a külső, a MULTP makróhívást tartalmazó makrónak is formális paramétere.

A végrehajtandó feladatok:

— az X3 helyén a formális-aktuális paramétercsere elvégzése az első szint paramétertábla-bejegyzésének alapján;

— a címkemezőben szereplő „CIM2" relatív szimbólum elhelyezése az első szintű makrócímketáblában (ez tehát még a külső makró címkéje lesz);

— a műveletikód-mezőben szereplő makrónév felismerése, a definiciós táblából a makró elhelyezkedési kezdőcímének meghatározása;

— az első szinthez tartozó makrócímketábla ideiglenes lezárása és kivitele a verem megfelelő eleme által meghatározott kezdőcímű lemezterületre;

— a makrókifejtési verem mélyítése: a megszakítás helyének és az aktuális paramétereknek az elhelyezése a megszakítási hely és a makróparaméterek veremtárában, LMCT növelése és az új érték beírása a makrócímketábla verembe;

— a nyert elhelyezkedési kezdőcímmel mint bemenő változóval a 3. operátor indítása és az SPB1-en az első beolvasott makrósor megjelenésének kivárása;

— a sor megőrzése a második menet számára.

A makrókifejtési veremtárak állapotát a vizsgált sor feldolgozása előtt a *3. ábra* adja meg. Az ábra szerinti felvételből kitűnik, hogy:

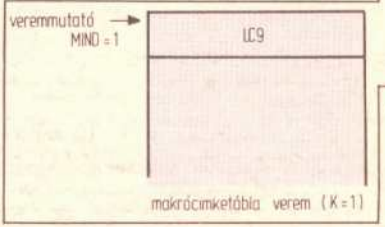
— a megszakítási hely és a makróparaméterek veremben a megszakítási cím helyén FFFF H áll, jelezve, hogy az elsődleges bemeneti periféria nem mágneslemez (különben érvényes lemezcím került volna erre a helyre);

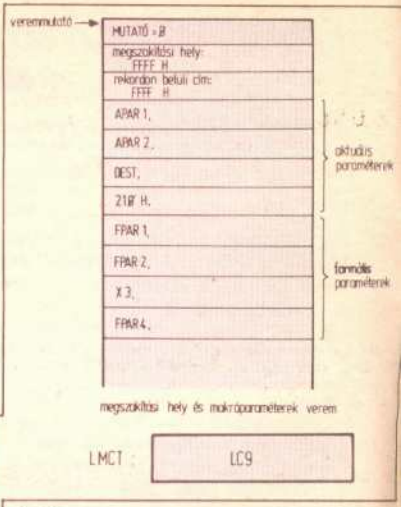
— az első szinten kifejtett makrónak négy formális paramétere van, köztük „X3", ami egyben a második szinten hívott makrónak is formális paramétere;

— az aktuális paraméterek között szimbolikus nevek mellett számkonstans is szerepel.

A rendszer működését újból a kapcsolatábra ugyanattól a pontjától kezdődően figyeljük. Az indított operátorok (az előző példánál kevésbé részletezve):

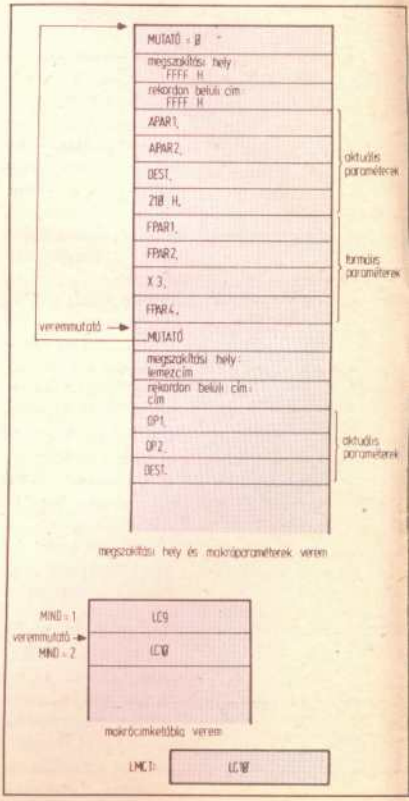
Az operátor típusa		Az operátor funkciója
5	—	Mezőkre bontja a feldolgozandó sort.
6	—	Nem megjegyzés sor: T.
6	—	IF2=Ø: T.
6	—	DEF=Ø: T.
7	—	Elvégzi a paramétercserét: a hívósorban egyetlen olyan szimbólumot talál, amely egyezik egy, a formális paraméterek között szereplő szimbólummal („X3"), ezt a neki megfelelő (harmadik) aktuális paraméterre cseréli ki („DEST"), mégpedig mind az operandusmezőben, mind az SPB2 pufferterületen. A sor végleges alakja tehát: CIM2 MULTP OP1,OP2,DEST lesz.
6	—	Van címke: F.
6	—	MIND=1: F.
8,9	—	A címkemező tartalmának feldolgozása megegyezik az 1. példában leírtakkal, azzal a különbséggel, hogy a bejegyzést nem a modulcímketáblában, hanem a kifejtés alatt álló makróhoz rendelt címketáblában helyezi el.
12	—	Az operátor a műveletikód-mező tartalmával (MULTP) megegyező szimbólumot keres a makródefiníciós táblában. Feltételezésünk szerint talál ilyen szimbólumot, s a táblabejegyzést hozzáférhe-





3. ábra

4. ábra



tövé teszi a T ágon következő operátor számára.

13 — Az operátor indítja a 2. szintű makrókifejtést, azaz

- az LC9-es lemezcímen kezdődő területre kiviszi a makrócímketábla ideiglenesen lezárt rekordját;
- a soron következő lemezterület kezdetére állítja az LMCT változót: LC1Ø = LC9 + 1 (K = 1);
- mélyíti a megszakítási hely és makróparaméterek veremtárát (beírja az első szintű kifejtés megszakítási helyét és a makróhívás aktuális paramétereit);
- mélyíti a makrócímketábla vermet az LC1Ø cím beírásával;
- a makródefiníciós táblából kiolvassa a MULTP makró definíciójának elhelyezkedési kezdőcímét és a 3. operátor meghívásával beolvassa SPB1-re annak első sorát;
- MIND értékét 2-re növeli.

A két veremmemóriának az operátori működés befejeződése után előálló állapotát a *4. ábra* mutatja be. A hívósor lemezre kerül, lezáródik a sorfeldolgozási ciklus, s indulhat a következő sor feldolgozása, amelynek tartalma éppen MULTP MACRO X1,X2,X3 lesz. A második szinthez tartozó formális paraméterek csak ekkor kerülnek be a veremtárba.

Sorozatunkban a (cross-)assembler programok működésének egy lehetséges algoritmusát tárgyaltuk rendszermodellezési eszközökkel. Reméljük, az olvasó meggyőződhetett a alkalmazott módszer hasznosságáról.

VÁRGEDŐ TAMÁS

COMMODORE—64

BASIC RAM felosztás

A C64 memóriájának 64 k RAM-ja van. Ez azonban nem jelenti azt, hogy a felhasználónak a BASIC program tárolására ennyi szabad terület áll rendelkezésére. A 0—2048-ig terjedő területet az operációs rendszer a rendszer változóinak tárolására, munkaterületeknek és a képernyőtárnak lefoglalja.

A 40960 feletti területen a RAM-on kívül ugyanazokon a címeken RAM-ok is vannak, amik a gép operációs rendszerét és az alap karakterkészletet tárolják. Ezt a RAM területet a BASIC nem használja. Ide helyezheti el a felhasználó saját karakterkészletét, saját gépi kódban írt programjait és saját szerkesztésű adatait.

A BASIC munkaterület tehát alaphelyzetben, a gép bekapcsolása után a 2049—40960 közötti terület. Itt fognak elhelyezkedni a BASIC programok, a program által használt változók és sztringek. A megmaradó terület a szabad BASIC terület.

A teljes munkaterület felosztását a gép operációs rendszere végzi, de lehetőség van a felosztás megváltoztatására. A felosztást a munkaterület mutatóinak olvasásával vizsgálhatjuk, azok átírásával pedig megváltoztathatjuk. A mutatók a következő tárcímeken helyezkednek el *(2. ábra)*:

43-44 SOB (Start Of Basic)
BASIC terület eleje, alaph: 2049
45-46 SOV (Start Of Variables)
változók kezdete, alaph: 2051
47-48 SOA (Start Of Arrays)
tömbök kezdő címe, alaph: 2051
49-50 EOA (End Of Arrays)
tömbök vége, alaph: 2051
51-52 BOS (Bottom Of Strings)
sztringterület, alja, alaph: 40960
55-56 TOM (Top Of Memory)
BASIC terület vége, alaph: 40960

A „Programmer's Guide" részletesen ismerteti a memória mutatóinak használatát, és megemlíti azok áthelyezésének lehetőségét. Az áthelyezett „SOV" (Start Of Variables = változók kezdete mutató) esetén a program már nem módosítható, mivel a program vége és a „SOV" nem esik egybe. Az ilyen programmal végzendő manipulációhoz ismerni kell a program végcímét. Ezt a problémát oldja meg az itt közölt mintaprogram.

A program írásakor vagy betöltésekor a program elhelyezése a SOB által meghatározott címtől kezdődik. A program hosszától függően a SOV mindig följebb tolódik úgy, hogy a program által elfoglalt terület végét mindig követi. Ennek megfelelően változik a többi mutató is.

Mint látjuk, minden mutató két bájtot foglal el. Az első a cím alacsonyabb, a második a magasabb helyértékű bájtját tartalmazza. Például a SOV értékét így kapjuk meg:

```
PRINT PEEK(45)+PEEK(46)*256
```

Valamelyik mutatóba új értéket a következő módszerrel írhatunk be (UJ = új érték):

```
HI=INT(UJ/256):LO=UJ-HI*256
POKE 45,LO:POKE 46,HI
```

A mutatók átállítását azonban nagy körültekintéssel kell végezni, hiszen ezek jelzik, hogy hol kezdődik és végződik a programunk, hol kezdődnek és végződnek a különböző típusú adataink (változók, tömbök, sztringek). A mutatók átállításával hozzáférhetetlenné tehetjük azokat a gép számára. A BASIC NEW parancs sem törli valójában a meglévő programot, hanem csak a mutatókat állítja vissza a SOB értékére. A BASIC CLR parancs a SOB-ot és a SOV-ot változatlanul hagyva, csak a többi mutatót állítja; a program működőképes marad, de az eddig előállított változók eltűnnek.

Mi célja lehet e mutatók megváltoztatásának? Például, hogy a BASIC programtól független területeket foglaljunk le a tárban

1. ábra

egyéb célokra, sprite-ok, gépi kódú programok vagy adatok tárolására.

Sprite-ok

A sprite-okat a tárnak abban a 16 k-s blokkjában kell elhelyezni, amelyikben a képernyőtár is van. A képernyőtár alaphelyzetben az 1024—2023 közötti címen van, így a sprite-oknak is az első 16 k-s blokkban kell lenniük. Elvileg így a 2048—16383-ig terjedő területből választhatnánk, de a valóságban csak a 2048—4095 és a 8192—16383 közötti terület használható. Mivel a BASIC program is ezen a területen dolgozik, így vagy a program fogja elrontani a sprite-okat, vagy azok rontják el a programot. Ezt megelőzhetjük, ha a tárnak egy részét lefoglaljuk.

Gépi kódú programok

Ha munkánkhoz gépi kódú programot is használunk, azt elhelyezhetjük a 40960 fölötti területen. Szükség lehet arra is, hogy a gépi kódú rész a BASIC programmal egy egységet alkosson, és azzal együtt lehessen betölteni. Ebben az esetben szintén le kell foglalni egy területet.

A BASIC területen belül egy meghatározott terület lefoglalására három lehetőség van: területfoglalás a BASIC program előtt, után, vagy a BASIC munkaterület végén.

Területfoglalás a BASIC program előtt

A SOB feljebb helyezésével tetszés szerinti terület foglalható le a BASIC program elhelyezési területe előtt *(2. ábra)*.

Például a SOB-ot 8192-re állíthatjuk a következő módon: POKE 43,0:POKE 44,32:POKE 8191,0:NEW

A POKE 8191,0 az új SOB előtti bájtot állítja nullára. Jegyezzük meg, hogy a BASIC kezdőcím előtti bájtnak mindig nullának kell lennie, ellenkező esetben a gép SYNTAX ERROR-t jelez. A NEW parancs a többi mutatót igazítja a megváltozott SOB-hoz.

A módszer hátránya, hogy az utasítást a program betöltése előtt kell kiadni. Ha az áthelyezett SOB után kezdjük el betölteni a programot, akkor az a tárban a 8192-es címtől kezdve helyezkedik el, szabadon hagyva a 2048—8190 közötti területet.

Területfoglalás a BASIC program végén

A program betöltésekor vagy beírásakor a program a SOB-tól kezdődően fog a tárban elhelyezkedni. A program végét a SOV jelzi. Innen kezdve helyezkednek el a változók. Ha a SOV-ot feljebb helyezzük még mielőtt bármilyen változót használnánk, a program vége és a változók kezdete előtt tetszés szerinti területet foglalhatunk le. Amit az így lefoglalt területre helyezünk, az a program futása alatt sértetlen marad. De ennél sokkal nagyobb a jelentősége annak, hogy a BASIC SAVE a mentést a SOB-tól kezdve a SOV-ig végzi, így a BASIC programmal együtt a lefoglalt terület tartalmát is kimenti. Ugyanígy a LOAD parancs az egészet vissza fogja tölteni a tárba *(3. ábra)*.

2. ábra

Hátránya ennek a megoldásnak, hogy ha a BASIC programot és a lefoglalt területre töltött gépi kódú programot egyszer így összetársítottuk, akkor a BASIC programban már semmilyen változtatást nem lehet végezni. Ennek oka a következő. A BASIC program végét a SOV már nem helyesen jelzi. A BASIC program módosításakor a SOV értéke annyival fog változni, ahány bájtot a programhoz hozzátettünk vagy töröltünk. Ha ismerjük a SOV előző és a megváltozott értékét is, akkor abból kiszámíthatjuk, hogy szabad területünk helye hogyan változott. Többszöri változtatgatások után előfordul, hogy nem tudjuk követni azokat. Ha ilyenkor megkeressük a BASIC program végét, akkor ismét megkaphatjuk a szabad terület helyét: a program vége és a SOV közötti területet.

A *P—1 jelű program* a BASIC program által elfoglalt terület utolsó bájtját keresi meg és írja ki a képernyőre. A keresés a következő módszerrel történik. A BASIC utasításokat a tárban egy-egy 0-ás bájt választja el. Ezek után egy kétbájtos mutató következik, ami a következő BASIC utasítás tárbeli kezdőcímét mutatja. Ha nincs több utasítás, akkor ez a két bájt is nulla. Ezt az egymás utáni három nullás bájtot keresi meg a program.

A programot saját programunk elé kell beírni. A saját program ez után, a 10-es sorszámtól kezdődik. A saját program indítása a RUN paranccsal, a BASIC végét kereső program indítása pedig RUN 2 paranccsal történhet.

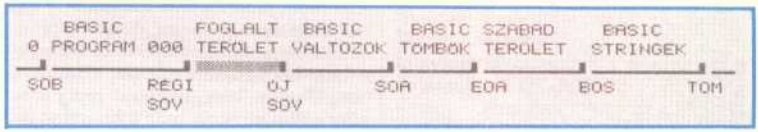
Ha a SOV-ot megváltoztatjuk és programunkat is módosítjuk, a program végét a RUN 2 paranccsal bármikor ismét megkaphatjuk, miután az végigköveti a BASIC sorok kezdetére mutató címeket, amíg a BASIC végét jelző 00 értékű mutatót meg nem találja.

A *P—2 jelű program* ugyanezt a feladatot látja el gépi kódú program segítségével.

A 100-as sorban történik a gépi kódú program betöltése.

A 110-es sor ellenőrzi, hogy helyes adatokat töltöttünk-e be. A 120-as sor törli a BASIC programot.

A program futtatása után vihetjük be saját programunkat, átírhatjuk a SOV-ot és módosíthatjuk programunkat. A BASIC program végét a SYS 49169 paranccsal írhatjuk ki.



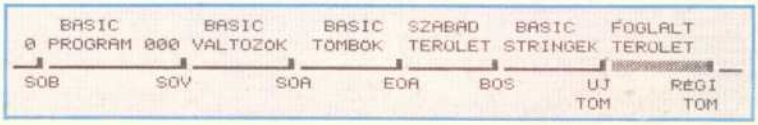
3. ábra

Területfoglalás a BASIC munkaterület végén

A BASIC munkaterület a SOB-tól a TOM-ig terjed. A program futása közben keletkező sztringeket a gép a TOM-tól lefelé helyezi el. Az elfoglalt terület alját a BOS jelzi. Ha a program indítása előtt a TOM-ot lefelé eltoljuk, akkor a régi és az új TOM közötti terület szabaddá válik. A TOM mutató lekérdezését és átírását a mutatóknál ismertetett módon végezhetjük *(4. ábra)*.

ZSOM BÉLA

4. ábra



BASIC és gépi kód

Legutóbb megnéztük egy rövid gépi kódú rutin elkészítésének folyamatát. Ez a kiszámított GOTO utasítás használatát lehetővé tevő alprogram a gép csak olvasható memóriájában található gépi kódú rutinokat aktivizálta JSR és JMP utasításokkal.

Most az úgynevezett töltőutasításokkal és néhány velük kapcsolatos tudnivalóval ismerkedhetünk meg.

A töltőutasítások

Nevük a „load" angol szó első és utolsó betűjéből és a használt regiszter nevéből áll. Hárman vannak: LDA, LDX, LDY. Használatuk az értékadó utasításokéhoz hasonlítható: az operandussal meghatározott memóriacím tartalma a műveleti kódban szereplő regiszterbe kerül. A regiszter korábbi tartalma elvész, a címzésben szereplő memóriabájt tartalma változatlan marad.

A töltőutasítások a mikroprocesszor állapotregiszterének N- és Z-bitjeit állítják be, a betöltött bájt értékének megfelelően. Ezeknek a feltételbiteknek a beállítását a töltőutasítást követő, ún. feltételes ugróutasításban lehet kihasználni. Ezekről később lesz szó.

A feltételbitek értelmezése

A regiszterbe töltött bájt tartalmát tekinthetjük egy olyan előjel nélküli számnak, melynek értéke a 0 ... 255 tartományba esik, de ugyanígy tekinthetjük egy —128 ... +127 értéktartományba eső előjeles számnak is. Ez utóbbi esetben, a szám 7. bitjét előjelbitként használva, az ún. kettes komplemens ábrázolással dolgozhatunk. Ebben az esetben segít az N-bit; a regiszterbe töltött szám előjelét megőrzi egy esetleges elágazó utasítás számára, mely a regiszterbe töltött szám pozitív vagy negatív voltától függően szabja meg a további programfutás menetét.

Röviden: az N-bit a töltőutasítások végrehajtása során az éppen használatos (A, X, ill. Y) regiszter 7-es sorszámú bitjének értékét veszi fel, vagyis akkor lesz 1, ha a regiszterbe kerülő szám negatív.

Egyszerűbb a Z-bit működésének megértése. Akkor veszi fel az 1-es értéket, ha a regiszterbe kerülő bájt minden bitje 0, vagyis a regiszterbe kerülő szám nulla (zéró).

A Z-bit állapotát is feltételes ugróutasításban lehet kihasználni.

Tárolóutasítások

A töltőutasítások párjai, a regiszterek tartalmát viszik (másolják) az operandus által meghatározott tárterületre. Nevük az angol „store" szó első két betűjéből és a használt regiszter nevéből tevődik össze. Szintén hárman vannak: STA, STX és STY. Működésük a töltőutasításokéhoz hasonlóan a magas szintű nyelvek értékadó utasításához hasonlítható. Egy nagyon fontos tulajdonságuk különbözteti meg őket a töltőutasításoktól: a tárolóutasítások a feltételbiteket változatlanul hagyják. Érdemes megjegyezni!

BARNA LÁSZLÓ

```
1  GOTO 7
2  PRINT : PRINT " A BASIC PROGRAM UTOLS
   O BYTE-JA:": PRINT
3  MUTAT = 43
4  MUTAT = PEEK(MUTAT) + PEEK(MUTAT +
   1) * 256
5  IF MUTAT > 0 THEN PRINT "[CUU]";M
   UTAT + 2: GOTO 4
6  END
7
```

A P—1 program

A P—2 program

```
10  PRINT "[CLR][CD][CD][CD]"
20  PRINT "        **********************"
22  PRINT "        *                    *"
24  PRINT "        *  [RVS] END OF BASIC
    [RVO]   *"
26  PRINT "        * <BAS.PROGR. VEGE> *"
28  PRINT "        *  HIVAS:SYS 49169  *"
30  PRINT "        *                    *"
32  PRINT "        **********************"
40
50  SOB = PEEK(43) + PEEK(44) * 256
60  PRINT "[CD][CD][CR][RVS]S[RVO]TART
    [RVS]O[RVO]F [RVS]B[RVO]ASIC:":
    PRINT "(BASIC PROGRAM KEZDOCIME)":
    PRINT SOB
100 FOR I = 49152 TO 49288: READ A:
    POKE I,A:S = S + A: NEXT
```

```
110 IF S <> 18828 THEN PRINT "HIBA
    A DATA SORBAN"
120 POKE SOB,0: POKE SOB + 1,0: SYS 49
    169: NEW
1000 DATA 32 , 69 , 78 , 68 , 32
1002 DATA 79 , 70 , 32 , 66 , 65
1004 DATA 83 , 73 , 67 , 32 , 65
1006 DATA 84 , 58 , 169 , 13 , 32
1008 DATA 210 , 255 , 162 , 0 , 189
1010 DATA 0 , 192 , 32 , 210 , 255
1012 DATA 232 , 224 , 17 , 208 , 245
1014 DATA 169 , 13 , 32 , 210 , 255
1016 DATA 169 , 43 , 133 , 252 , 169
1018 DATA 0 , 133 , 253 , 141 , 192
1020 DATA 2 , 160 , 0 , 177 , 252
1022 DATA 200 , 17 , 252 , 240 , 59
1024 DATA 177 , 252 , 133 , 255 , 136
1026 DATA 177 , 252 , 133 , 254 , 169
1028 DATA 13 , 32 , 210 , 255 , 169
1030 DATA 145 , 32 , 210 , 255 , 164
1032 DATA 254 , 165 , 255 , 32 , 145
1034 DATA 179 , 32 , 221 , 189 , 162
1036 DATA 0 , 189 , 0 , 1 , 8
1038 DATA 32 , 210 , 255 , 232 , 40
1040 DATA 208 , 245 , 173 , 192 , 2
1042 DATA 240 , 1 , 96 , 165 , 254
1044 DATA 133 , 252 , 165 , 255 , 133
1046 DATA 253 , 76 , 51 , 192 , 169
1048 DATA 1 , 141 , 192 , 2 , 101
1050 DATA 254 , 133 , 254 , 165 , 255
1052 DATA 105 , 0 , 133 , 255 , 76
1054 DATA 69 , 192
```

Z80 programozási

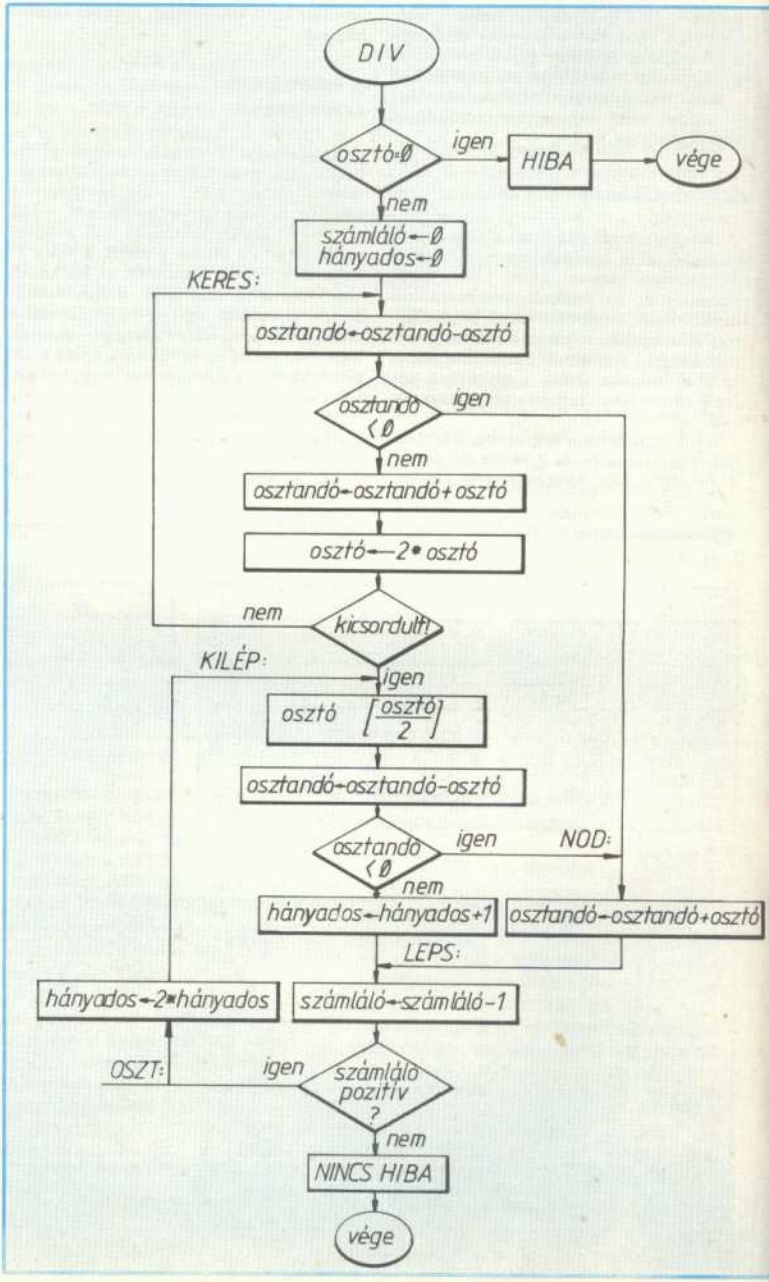
Az előző részben megnéztük, hogy hogyan tudunk két bájton ábrázolt számokat decimálisan kiíratni. Most nézzük meg ennek a fordítottját!

1. Írjuk szubrutint, amely egy, a HL által mutatott címtől karakter alakban adott decimális, előjel nélküli egész szám bináris értékét beírja DE-be! Ha hibás a szám, carry bit jelezze a hibát!

A karakter alakban adott decimális egész azt jelenti, hogy a tárban valahol a számjegyek ASCII kódjai szerepelnek egymás után. Például! 124 a következőképpen: ...,49,50,52,... Valahonnan tudnunk kell, hogy hol kezdődik és hol végződik a szám. Az első karakterre mutasson HL regiszterpár, és tartson addig a szám, ameddig számjegyek vannak, azaz az első nem "0" és "9" közé eső bájtig.

Egy tetszőleges számrendszerben felírt szám, amelynek jegyei: a, b, c, d (legyen most csak négyjegyű!), értéke aA^3+bA^2+cA+d, ahol A a számrendszer alapszáma. Ez felírható $((aA+b)A+c)A+d$ alakban is. Ez az alak, az ún. Horner-séma, egy nagyon egyszerű kiszámítási módot sugall: szorozzuk a-t A-val, adjuk hozzá b-t, szorozzuk A-val, adjuk hozzá c-t, szorozzuk A-val, adjuk hozzá d-t. Ha pedig a szám végére írnánk még egy számjegyet, akkor az előzőekből kiszámítható az így kapott szám értéke, csak szorozni kell A-val, és hozzáadni az újabb számjegyet. Az algoritmus ezek után a következő.

Mindaddig, amíg találunk számjegyet, az eddigi értéket szorozzuk tízzel, és hozzáadjuk az újabb számjegy értékét. Hogy ne legyen gondunk az „eddigi" értékkel, a szám elején feltételezünk egy fiktív nullát, amit azután szorozhatunk tízzel és hozzáadhatjuk a megfelelő számot (lásd az *1. programot*). A program elején DE-be nullát töltünk, feltételezve a fiktív nullát a szám előtt. Az akkumulátor vesszős párjában számláljuk a számjegyeket, ezért ezt a szubrutin elején lenullázzuk. Ha ugyanis az első nem számjegykarakter előtt nem találunk egyetlen számjegykaraktert sem, akkor visszatéréskor a carry bittel jeleznünk kell a hibát. Ezután előveszük a következő karaktert. Az akkumulátorból levonva "0"-t megkapjuk az ASCII kódhoz tartozó számot. Ha a kivonás után carry=1, akkor nem számjegykarakter volt az elővett karakter. Akkor sem, ha a kivonás után az akkumulátor nagyobb mint 9, azaz ha az eredeti érték nagyobb volt, mint "9". Ezután szoroz-



yakorlatok 3.

```
;**************************************
;*      DE-t és BC-t szorozza         *
;*      eredmény HL-ben               *
;**************************************
MULT   LD   HL,Ø
CIKL   SRL  B          ;BC<-
       RR   C          ;   BC/2
       JR   NC,NOPL    ;nem kell hozzá adni
       ADD  HL,DE
       RET  C          ;tulcsordulás
NOPL   EX   DE,HL      ; DE<-
       ADD  HL,HL      ;   DE*2
       EX   DE,HL      ;
       JR   C,VEG      ;hiba ha BC nem Ø
       LD   A,B
       OR   C          ;----------------+
       JR   NZ,CIKL    ;ha BC nem nulla !
       RET             ;carry=Ø <-------+
VEG    LD   A,B
       OR   C          ;carry
       SUB  1          ;    beállitása
       CCF             ;      megfeleloen
       RET
```

1. program

```
;**************************************
;* DE<- karakteresen megadott szám    *
;*      a szám elejére HL mutat       *
;*      a szám vége az első nem számjegy *
;*                          karakter  *
;**************************************
VAL    XOR  A          ;Töröljük
       LD   D,A        ;  DE-t
       LD   E,A        ; és A'-t
       EX   AF,AF'
NXCH   LD   A,(HL)
       INC  HL         ;következő karakter
       SUB  '0'        ;konverzió
       JR   C,EON      ;nem számjegy
       CP   10
       JR   NC,EON     ;nem számjegy
       CALL TEN        ;szorzás tízzel (2.4 prg.)
       RET  C          ;tulcsordulás
       LD   C,A        ;BC-be a számjegy
       LD   B,Ø        ;         értéke
       EX   DE,HL      ; DE<-
       ADD  HL,BC      ;  DE*  ; a szám plusz
       EX   DE,HL      ;   BC  ; az aktuális számjegy
       RET  C          ;tulcsordulás
       EX   AF,AF'     ;számjegyek
       INC  A          ; számlálása
       EX   AF,AF'
       JR   NXCH       ;következő karakter
EON    EX   AF,AF'
       SUB  1          ;carry beállítása
       RET
```

2. program

3. program

```
;**************************************
;*      Osztó rutin                   *
;*                                    *
;*      BC<-HL/DE                     *
;*                                    *
;**************************************
DIV    LD   A,D
       OR   E          ;nullával
       SCF             ;  való
       RET  Z          ;    osztás
       XOR  A          ; számláló     ;A
       LD   B,A        ;  és hányados ;BC
       LD   C,A        ;         törlése
KERES  OR   A          ;carry törlése
       SBC  HL,DE      ;osztandó<-osztandó-osztó
       INC  A          ;számlálás
       JR   C,NOD      ;nagyobb lett az osztó
       ADD  HL,DE      ;helyreállítás
       EX   DE,HL      ;osztót
       ADD  HL,HL      ; szorozzuk
       EX   DE,HL      ;  kettővel
       JR   C,KILEP    ; ha nem fér el
       JR   KERES
OSZT   SLA  C          ;hányadost
       RL   B          ;  szorozzuk kettővel
KILEP  RR   D          ;osztót osztjuk
       RR   E          ;          kettővel
       SBC  HL,DE      ;nem volt carry
       JR   C,NOD
       INC  C          ;hányados növelése
       JR   LEPS
NOD    ADD  HL,DE      ;visszaállítás
LEPS   DEC  A          ;számláló csökkentése
       JR   NZ,OSZT
       OR   A          ;CY<-Ø
       RET
```

zuk tízzel, és hozzáadjuk az aktuális számjegyet HL-hez, amit előzőleg már megcseréltünk DE-vel. Ha bármelyik művelet után carry = 1, akkor hiba van (túlcsordulás). Visszacseréljük HL-t és DE-t, majd növeljük A' regisztert, így A' a feldolgozott karakterek számát jelenti. Az EON címkénél a SUB utasítás hatására A' a feldolgozott karakterek száma mínusz egyet fog mutatni, és carry akkor lesz egy, ha A' nulla volt, azaz ha nem volt számjegy. (DEC A nem használható, mert nem álllítja a carry bitet.)

Mit ad eredményül a program, ha a pufferben 256 darab "Ø"-t talál? Erről az esetről azért nem kell gondoskodnunk, mert a gyakorlatban előforduló pufferek rövidebbek.

Írjunk olyan szubrutint, amely figyelembe veszi az előjelet is!

A kétbájtos egész aritmetika két legnehezebb problémája a szorzás és az osztás. Foglalkozzunk most ezzel a két problémával!

2. Írjunk szubrutint, amely két előjel nélküli egész számot összeszoroz! Legyen a szorzó BC, a szorzandó DE, és képezzük az eredményt HL-ben!

Az előző részben a 4. program a tízzel való szorzás volt. Most úgy kell szorozni, hogy nem tudjuk előre, hogy mivel szorzunk. Az algoritmus a következő. HL-be nullát töltünk, BC-t osztjuk kettővel, és ha páratlan volt az osztás előtt, azaz a léptető-utasítások után carry = 1, akkor HL-hez hozzáadjuk DE-t, majd DE-t szorozzuk kettővel, majd ismét osztjuk BC-t, és így tovább mindaddig, míg BC nem nulla. A ciklus n-edik lefutása után DE-ben az eredeti érték 2-szerese, BC-ben az eredeti érték 2-ed része van, azaz ekkor BC páratlan, ha az eredeti érték n-edik bitje egyes, és páros, ha nulla volt. Így ha BC eredeti értéke $2 + 2^2 + 2^3$, akkor DE eredeti értékének $2, 2^2, 2^3$-szeres értékei adódnak össze HL-ben. *(2. program)*

Néhány szó a regiszterek megválasztásáról. Mindenképpen HL kellett hogy legyen a gyűjtő regiszter, mert más regiszterpárhoz, IX és IY kivételével, nem tudjuk közvetlenül hozzáadni egy másik regiszterpár értékét. A szorzandókra BC és DE maradt. Szerepük hasonló, mégsem mindegy, hogy melyiket léptetjük felfelé és melyiket lefelé. BC-t mind felfelé, mind lefelé csak két, egyenként kétbájtos utasítással lehet léptetni, ahogy a program listájában is látni lehet. A DE regiszterpár balra léptetésére, azaz kettővel való szorzására jó lehetőséget nyújt, hogy a HL regiszterpárral könnyen megcserélhető, és így négy helyett csupán három bájtot foglal el az utasítás, sőt egy kicsit gyorsabb is.

Előjeles számokat például úgy is lehet szorozni, hogy az abszolút értékeket szorozzuk, majd az előjel szerint, amit előre ki kell számítani, ha kell, mínusz egyszeresére váltjuk a számot. Szorzásra nem mindig ez a legmegfelelőbb algoritmus. Például, ha az akkumulátor tartalmát akarjuk 255-tel szorozni, akkor sokkal célravezetőbb 256-tal szorozni, ami elég egyszerű, és levonni az eredeti értéket.

3. Írjunk szubrutint, amely kiszámítja két 16 bites szám hányadosát! (Lásd a folyamatábrát!)

A legegyszerűbb módszer az osztásra az, hogy addig vonjuk ki az osztót az osztandóból, amíg az nagyobb nála, és megszámoljuk, hogy ezt hányszor tettük meg. Előfordul, hogy tényleg ez a legcélravezetőbb, de egy általános osztóprogramban semmiképp. Gondoljunk csak arra, hogy mi időbe telne 60 000-et elosztani eggyel!

Nem az osztót vonogatjuk le tehát az osztandóból, hanem annak valamely kettőhatványszorosát. Azért éppen kettőhatványszorost, mert azt könnyen elő tudjuk állítani. Nem egyesével számlálunk, hanem mindig a megadott kettőhatvánnyal növeljük a számláló értékét, ahol az eredményt kapni akarjuk. A kettőhatvány, amivel megszorozzuk az osztót, legyen a lehető legnagyobb úgy, hogy még éppen ne váljon nagyobbá, mint az osztandó. Ha levonjuk az így kapott értéket az osztandóból, akkor az most már kisebb lesz, mint az osztó, ezért azt osztjuk kettővel, és ha így kisebbé vált az osztandónál, akkor levonjuk, ha nem, akkor nem, és újból osztjuk kettővel, és így tovább mindaddig, míg az osztó el nem éri az eredeti értékét. Végül az osztandó helyén az osztó által alkotott maradéka marad. A számlálást úgy oldhatjuk meg, hogy mindig eggyel növeljük az először lenullázott számlálót, és valahányszor osztjuk kettővel az osztót, a számlálót szorozzuk kettővel. *(3. program)*

A szubrutin először ellenőrzi, hogy az osztó nem nulla-e. Ez az egyetlen hibalehetőség. Túlcsordulás másutt nem léphet fel, hiszen a hányados semmiképpen nem lehet

nagyobb, mint az osztandó. Ezután lenullázzuk az akkumulátort és a BC regiszterpárt. A BC regiszterpárban fogjuk képezni a hányadost, az akkumulátorban pedig azt számoljuk, hogy hányszor szoroztuk kettővel az osztót, míg az éppen nagyobbá vált, mint az osztandó.

Mivel így eggyel többször szoroztunk kettővel, mint ahogy arra szükségünk lenne, kivéve azt az esetet, amikor az osztó nagyobb, mint az osztandó, ezért a BC balra tolása után rögtön az osztó kettővel való osztásával kezdjük a levonogatásokat.

BC-ben a hányadost szeretnénk kapni, ezért amikor balra toljuk, azaz kettővel szorozzuk, a művelet után nem lehet carry=1, hiszen az túlcsordulást jelentene; így az osztó kettővel való osztásakor a 15-ös bitbe nulla kerül, és nem okoz gondot, hogy forgató- és nem léptetőutasítást használunk. Ez után a művelet után ugyancsak nem lehet carry=1, mert DE, azaz az osztó alsó bitjeibe nullát töltöttünk, és nem forgatjuk annyiszor jobbra, hogy értékes bit kicsorduljon. A BC-ben először nulla van, így nem baj, hogy először azt szorozzuk kettővel.

Ha az osztó nagyobb, mint az osztandó, akkor nem történik műveletvégzés, a carry törlése után befejeződik a szubrutin. Amikor az osztót szorozzuk kettővel, az kétféleképpen válhat nagyobbá az osztandónál. Vagy úgy, hogy DE nagyobb lesz, mint HL, és a kivonás után carry bit jelzi ezt, vagy úgy, hogy nagyobbá válik, mint ami 16 biten elfér, ezt a kettővel való szorzás után jelzi a carry jelzőbit, és ekkor is biztos, hogy nagyobb az osztandónál, hiszen nagyobb minden 16 bites számnál. Ekkor a futás a KILEP címkével jelölt sorban folytatódik, és itt a 15-ös bitbe visszakerül a helyes érték. Ezért nem írhatunk ebbe a sorba léptetőutasítást.

Három különböző esetre nézzük végig a program futását! Egy olyanra, amikor az osztó nagyobb, mint az osztandó, egy olyanra, amikor az osztó nagyobb, mint 7FFFH, azaz a 15-ös bitje nem nulla, és kövessünk végig egy olyan esetet is, amelyik ezek egyike sem.

Gondoljuk végig a regiszterek megválasztását! Próbáljunk meg előjeles számok osztására programot írni! Próbáljunk meg olyan szorzó szubrutint írni, amely az eredményt 32 biten adja meg!

VERHÁS PÉTER

Egyváltozós függvények ábrázolása és megoldása

A program egyváltozós függvényt, illetve első deriváltját ábrázolja a megadott intervallumban. Megkeresi ebben az intervallumban a függvény lokális szélsőértékeit, és mint egyenletnek a gyökeit. A függvényt az 500-as sor tartalmazza, y=f(x) alakban.

Bemenetek

MODE 1	csak a függvényt, MODE 2 deriváltját is ábrázolja
k	kezdőpont
v	végpont (az ábra intervalluma)
F	függvényazonosító, nyomtatásra kerül
y	az ábrázolt y tengely hossza (egységben)
C1	a függvény színe
C2	a derivált színe ábrázolásnál
ITV	gyökkeresés léptetési intervalluma
IT	szélsőérték-keresés léptetési intervalluma

A program felépítése

5—10	kezdőértékek
10—40	értékkészlet AC-ban, min. és max. „P" és „O"-ban
40—50	bemenetek és nyomtatás
50—60	függvény és deriváltjának ábrázolása
69—80	x tengely
81—86	x tengely beosztások
87—100	y tengely
101—107	y tengely beosztások
110—170	gyökkeresés
180—230	szélsőérték-keresés
250	derivált függvény
300—310	x és y tengely beosztásainak kezdőértéke és léptetése
400—420	értékkészlet max. és min. kiválasztása
500	y=f(x) függvény

Egy harmadfokú (*1. ábra*) és két összetett trigonometrikus függvényen (*2. ábra*) mutatjuk be a program használatát. Az eredeti ábrák önmagukban érthetők voltak, de a négyszínű ábrázolás lehetőségével nem rendelkezünk, ezért tudni kell, hogy minden ábrán f(x) a lejjebb levő.

MÉSZÁROS CSABA

PTA–4000

```
  5:CLEAR :TEXT :P=9E99:O=-P:N#="##
    .######^^":INPUT "MODE (1,2) =";R
 10:O=51:DIM A(O):INPUT "K=";K,"U=";
    U:W=(U-K)/O:X=K-W:FOR I=0TO O:X=
    X+W:GOSUB 500
 20:A(I)=Y:GOSUB 100
 25:IF I=0OR R<>2GOTO 40
 30:GOSUB 250:GOSUB 400
 40:NEXT I:INPUT "F=";A$:USING N#:
    LPRINT "Y=";A$:LPRINT " MIN=";P;
    " MAX=";O:USING
 45:T=200:INPUT "Y=";E,"C 1=";C:
    COLOR C
 46:A$="F(x) ---":IF R=2LPRINT A$:
    INPUT "C 2=";D:COLOR D:LPRINT A$
    :GRAPH :SORGN :LINE (10,33)-(10,
    20):TEXT
 47:LF 2:GRAPH :GLCURSOR (5,8):SORGN
    :U=O-P:M=-P*E/U-E
 58:COLOR 8(L+3):FOR I=LTO O:X=I*T/O
    :Y=A(I):IF L=1GOSUB 250
 52:X=I*T/O:Y=Y*E/U+M
 55:IF I=LGLCURSOR (X,Y)
 60:LINE -(X,Y):NEXT I:IF R=2AND L=8
    LET L=1:GOTO 58
 65:Y=-O*E/(O-P):IF Y>8LET Y=5
 70:IF Y<-ELET Y=-E-5
 80:T=210:COLOR 8:LINE (-5,Y)-(T,Y):
    RLINE (-5,5)-(5,-5)-(-5,-5):
    GLCURSOR (T-6,Y-8):CSIZE 1:
    LPRINT "X"
 81:T=200:C=W-K:D=K:GOSUB 300
 82:B=(J-K)*T/(U-K):IF J=86GOTO 86
 83:LINE (B,Y+2)-(B,Y-3):L=B-3*LEN
    STR# J:IF L<-5LET L=-5
 84:GLCURSOR (L,Y-12):LPRINT STR# J
 86:J=J+I:IF J<=UGOTO 82
 87:X=K*T/(K-U):IF X<-5LET X=-5
 90:IF X>TLET X=T+5
100:LINE (X,-E-15)-(X,15):RLINE (-5,
    -5)-(5,5)-(5,-5):LPRINT "y":
    GLCURSOR (X+2,Y-8):LPRINT "0"
101:C=U:D=P:GOSUB 300:L=(LEN STR# J
    )*6+X+5)211)
102:B=(J-P)*E/U-E:IF J=86OTO 107
103:IF B>86OTO 105
104:LINE (X-2,B)-(X+3,B):IF L=8
    GLCURSOR (X+5,B-3):GOTO 106
105:GLCURSOR ((X-(LEN STR# J)*6-2),B
    -3)
106:LPRINT STR# J
107:J=J+I:GOTO 102
109:GLCURSOR (8,-E):TEXT :CSIZE 2:LF
    3
110:USING N#:INPUT "ITV.=";W:M=K-W
120:M=M+W:C=M:IF C>UGOTO 180
130:X=C:GOSUB 500:U=Y:X=X+.001:GOSUB
    500:D=C:C=D-U/(Y-U)*.001:IF C>M+
    WOR C<M6OTO 120
140:IF ABS (D-C)>1E-86OTO 130
170:LPRINT "GYOK=";C:GOTO 120
180:INPUT "IT=";W:X=K:A$=" max=":B$=
    " min="
190:B=W:GOSUB 500:U=Y:X=X+B:GOSUB 50
    0:Z=(U)Y):X=X-B
200:X=X+B:GOSUB 500:IF Z<>(U)Y)OR U=
    YLET B=-B/3:IF ABS B<1E-86OTO 23
    0
210:IF X+W>ULF 4:END
220:W=Y:GOTO 200
230:LPRINT B$(Z+1);(U+Y)/2;"   X=";X
    :GOTO 130
250:Y=(A(I)-A(I-1))/W:RETURN
300:A=10^(-INT (LOG (C/6))):I=INT (A
    *C/5+.5)/A:J=INT (D/I+.9)*I
310:S=J-I/2:IF D/S).85AND INT (A*S)/
    A=SLET J=S
320:RETURN
400:IF Y<PLET P=Y
410:IF Y>OLET O=Y
420:RETURN
500:Y=X*X*X+X*X-28*X:RETURN
```

Tártartalom a képernyőn

A bemutatott program a mikroszámítógép memóriájában tárolt információkat jeleníti meg. C64 gépre készült, de más géptípusra is könnyen átírható. A képernyőn egyszerre 16×8=128 bájt hexadecimális és karakteres alakját tanulmányozhatjuk.

A gyorsabb futás érdekében a hexadecimálisan megjelenő szubrutin a program elején szerepel. Ugyanezért a konstansokat változókban tároljuk, és a változóknak a program elején a felhasználási gyakoriságuk sorrendjében értékadásokkal helyet foglalunk. A valódi változók itt csak egy üres értéket kapnak.

A program valódi változói

Y: egy 8 bájtos memóriaszelet kezdőcíme
X: az átalakítandó és megjelenítendő érték
K: sorok száma a képernyőn
I: ciklusváltozó
Z: egy hexadecimális jegy értéke az átalakításkor

Lényeges konstansok

4096, 256, 16: a hexadecimális átalakításhoz szükséges osztók
0,1, ... ,F: hexadecimális karakterek, az A$ tömbben tárolódnak
127: maszk egy bájt előjelbitjének elnyomására
32: az ennél kisebb bájtok karakterképe helyett a tizedespont jelenik meg, melynek ASC kódja 46
8: ahány bájtról egy képernyősorban információ jelenik meg
65528: ennél nagyobb kezdőcímű 8 bájtos sorozatról már nem jeleníthető meg információ
T$: 37 darab léptetés a kereten belüli törléshez

Vezérlő és grafikus karakterek

CHR$(3):	STOP
CHR$(13):	RETURN
CHR$(17):	CRSR DOWN
CHR$(18):	RVS ON
CHR$(19):	HOME
CHR$(29):	CRSR RIGHT
CHR$(147):	CLR
CHR$(149):	barna
CHR$(194) és CHR$(221):	függőleges vonal karakterek

A program felépítése

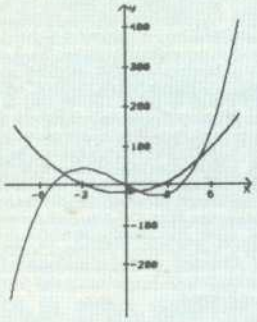
200—230: X megjelenítése 4 jegyű hexadecimális formában
220—230: X megjelenítése 2 jegyű hexadecimális formában
300—490: helyfoglalás a valódi változóknak és a konstansok áttöltése változókba (konverzió!!)
510 : képernyőszín-beállítás
520—540: kezdőcímbekérés
550—610: keret rajzolása
630 : STOP/RESTORE tiltás
640—700: 8 darab bájt megjelenítése
710 : a következő képernyősorra lépés, ha még nem telt be a képernyő
730—770: üzenet kiadása a képernyő alján
780—800: RETURN-ra várakozás, STOP esetén befejezés
810—850: törlés, és folytatás tiszta képernyőn

Speciális utasítások

POKE 198,0 : billentyűzet-puffer „ürítése"
POKE 808,225: STOP/RESTORE tiltás
POKE 808,237: STOP/RESTORE engedélyezés
POKE 781,X: POKE 782,N—1: SYS 59905 az X-edik sor elejétől N darab pozíció törlése a képernyőn, de a kurzor eredeti helye nem változik (N<128)

DR. SZÖRÉNYI MIKLÓS

```
Y=X**X+X*X-20*X
 MIN=-2.880000E 02
 MAX=+4.159999E 02
f(x)  ---
f'(x)  ---
```

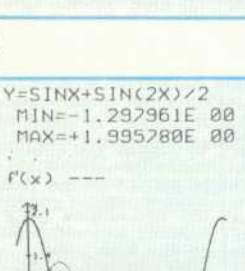


```
GYOK=-5.000000E 00
GYOK=+0.000000E 00
GYOK=+4.000000E 00
 max=+4.203149E 01
   X=-2.936724E 00
 min=-2.855001E 01
   X=+2.270096E 00
```

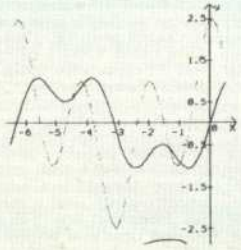
1. ábra

2. ábra

```
Y=SINX+SIN(2X)/2
 MIN=-1.297961E 00
 MAX=+1.995780E 00
f'(x)  ---
```



```
Y=SINX+SIN(3X)/2
 MIN=-2.488505E 00
 MAX=+2.487780E 00
f(x)  ---
```



```
100 GOTO 300
110 :
200 Z=INT(X/C)  X=X-Z*C   PRINT A$(Z);
210 Z=INT(X/B)  X=X-Z*B   PRINT A$(Z);
220 Z=INT(X/A)  X=X-Z*A
230 PRINT A$(Z)A$(X)" ";  RETURN
240 :
300 X=0  Z=0  I=0  A=16  E=127
310 F=32  S=0  Q=46  Y=0  K=0
320 B=256  H=8  C=4096  D=65528
330 P=128  Q=1  P=7  T=0  XR=781
340 YR=782  RS=808  KN=198
350 U=225  V=237  W=19  L=119
360 A$=CHR$(16)  B$=CHR$(29)
370 C$=CHR$(19)  D$=C$+CHR$(17)
380 E$=CHR$(147)  F$=CHR$(149)
390 T$="                                    "
400 T$=T$+"                                 "
410 P$=CHR$(3)  R$=CHR$(13)
420 K$=CHR$(221)+T$+CHR$(194)
430 H$="FOLYTATAS  'RETURN'"
440 N$=" DISPLAY MEMORY "
450 DIM A$(15)
460 FOR I=0 TO 9
470 A$(I)=CHR$(48+I)  NEXT I
480 FOR I=10 TO 15
490 A$(I)=CHR$(55+I)  NEXT I
500 :
510 POKE 53280,13  POKE 53281,13
520 PRINT E$F$;
530 INPUT "KEZDOCIM ( 0 - 65528 )  ";Y
540 IF Y<0 OR Y>INT(Y) OR Y>0 GOTO 520
550 PRINT E$SPC(11)A$N$
560 PRINT "┌──────────────────────┐";
570 PRINT "└──────────────────────┘"
580 FOR I=1 TO 16
590 PRINT K$  NEXT I
600 PRINT "└──────────────────────┘";
610 PRINT "┌──────────────────────┐"
620 :
630 POKE RS,U  PRINT D$
640 IF Y>0 THEN POKE RS,V  END
650 X=Y  PRINT B$;  GOSUB 200  S=Y+R
660 FOR I=Y TO S  X=PEEK(I)
670 GOSUB 220  NEXT I  PRINT A$;
680 FOR I=Y TO S  X=PEEK(I) AND E
690 IF X<F THEN X=G
700 PRINT CHR$(X);  NEXT I  PRINT
710 Y=Y+H  K=K+Q  IF K<A GOTO 640
720 :
730 PRINT
740 X=Y-P  PRINT SPC(13);  GOSUB 200
750 PRINT "-";  X=Y-Q  GOSUB 200
760 PRINT  PRINT
770 PRINT SPC(9)H$;  POKE KN,T
780 GET X$  IF X$="" GOTO 780
790 IF X$=P$ THEN POKE RS,V  END
800 IF X$<>R$ GOTO 780
810 POKE XR,W  POKE YR,L  SYS 59905
820 PRINT D$  K=T
830 FOR I=2 TO 17
840 PRINT B$T$  NEXT I
850 PRINT D$  GOTO 640
```

A MINŐSÉGÜGY KÖZÜGY

A minőség helye a viselkedés kialakításában. A viselkedés minősége

A könnyebb érthetőség kedvéért célszerű, ha konkrét példát vizsgálunk. Vegyünk például egy gyárat, amely önálló (nem tartozik bele trösztbe, kartellbe, szindikátusba, egyesülésbe stb.). Ennek a gyárnak kell a lehető legjobb minőségű viselkedést, működést megvalósítania. Világos, hogy erre az esetek többségében reménye sem lehet, ha már hosszabb ideje a világ legrosszabb minőségű árúját kínálja a világ legmagasabb árán úgy, hogy más gyárak sokkal jobbat, sokkal olcsóbban szállíthatnak a potenciális vevőknek, akár még a Föld túlsó oldaláról is. (A viselkedésminőség tehát nemcsak a vásárlók, hanem a konkurensek földrajzi helyzetétől is függ.) Ha a gyár az összes konkurenciánál jobb minőségű árut akar előállítani, lehet, hogy az túl drága lesz, és így a nyakán marad, vagy veszteséggel kell eladnia. Nyilván egyik sem lehet legjobb viselkedés. A viselkedésminőség javítása érdekében engedni kell tehát a termék minőségéből.

Tegyük fel, hogy a minőség mérhető és a mérés eredménye egy valós számmal leírható. Ekkor a minőség mérőszáma osztva az árral, fontos jellemzője lesz *a terméknek*, amely bizonyos mértékig a gyár működésének minőségét is jellemzi. Hogy a működés minőségét hogy mérjük, hogyan jellemezzük, az nem tartozik az egyszerű feladatok közé. Jelenleg nincs birtokunkban (talán soha nem is lesz) olyan egzakt módszer, amellyel általában viselkedésminőséget jellemző függvényeket alkothatunk. Ha a rendszereknek — amelyeknek viselkedésével foglalkozunk — követés, utánzás, alkalmazkodás, approximáció a feladata, vannak eljárásaink a feladat megoldása minőségének meghatározására (természetesen egy időintervallumra vonatkozóan). Mivel e feladattípus nagyon általános, várható, hogy e módszerek széles körben használhatóknak bizonyulnak. Akkor azonban, amikor nem tudjuk, hogy mit kellene közelítenünk, utánoznunk, követnünk, e módszereknek nem sok hasznát vehetjük.

Meg kell tehát elégednünk azzal, hogy a gyár viselkedése minőségének mérésére *adott valamilyen módszer, amelyet készként kell elfogadni és felhasználni.* (Ez „politika" jellegű adottság.)

A gyár tehát gyárt valamit valamilyen minőségben és valamilyen áron. Tegyük fel, hogy az éves haszon maximalizálására törekszik, az adott termelési, raktározási stb. korlátain belül. Ez a feladat általában megoldható, ha ismeri, hogy mely minőségű áruból mennyit tud eladni. (Feltesszük, hogy ugyanolyan minőségű árut gyárt egész évben.) Ekkor van egy függvény, amely az áru minősége függvényében a hasznot mutatja. Meg kell keresni a maximális haszonhoz tartozó minőséget (minőségeket), van tehát „optimális" minőség, amellyel a legnagyobb haszon elérhető. (Az „optimum" fogalma a gyakorlatban *sok szempontból problematikus. Semmiképp sem egzakt fogalom.* A matematikai „optimum"-ok közönséges extrémumok. Az extrémum elméletileg és gyakorlatilag is egzakt fogalom.) Ez az „optimális minőség" nem biztos, hogy az elérhető legjobb minőség lesz.

Az eddigiekben túlságosan egyszerűsített világot írtunk le, a valóság bonyolultabb szokott lenni. Gondoljuk csak arra, hogy valamely gyár, ha tudna is túl jó cipőket, túl olcsón gyártani, megtenné-e? Az elnyűhetetlen és kényelmes lábbeliket az emberek évekig hordanák. (És évekig nem lenne szükségük arra, hogy újat vegyenek. A gyár túl jó és túl olcsó termékei iránt megcsappanna a kereslet.) *Túlságosan jó és túlságosan olcsó termékek hamar eltűnnek a piacról.*

Több törhetetlenüveg-találmány volt már. Mindegyik találmány gyártási jogát megvették a (törékeny) üveget gyártó üveggyárak, és továbbra is törékeny portékákat gyártottak és gyártanak. Mert holnap is meg akarnak élni . . . Ugye furcsa?

Abból indultunk ki, hogy gyárunk független. Ez azonban ritka mint a fehér holló. Hogyan viselkedjen tehát egy olyan gyár, amely nem szabadúszó, hanem egy (vagy több) „csapatban játszik"? E probléma sok szempontból rokon azzal, amikor a független gyár egyes egységeit vizsgáljuk olyan szempontból, hogy azok hogyan viselkednek, hogyan navigálnak. (A gyáron belüli egységeknek vannak kötelezettségeik, de bizonyos mértékű szabadságuk is van. És nyilvánvaló, hogy a gyár viselkedését részeinek viselkedése határozza meg.)

A nehéz (és valóságosabb) probléma „csapatok", (bizonyos függetlenséggel is rendelkező részrendszerekből összetett rendszerek) legjobb irányításának (viselkedésének) problémája. Mégpedig minden szinten. Hogyan viselkedjen például egy tröszt (vezetése), a trösztthöz tartozó vállalatok (vezetése), a vállalatokhoz tartozó gyárak (vezetése) a gyárakhoz tartozó gyáregységek, üzemek, munkahelyek stb. vezetése, és végül az egyes dolgozók? E fontos problémát sokan kerülgették már. Sokat beszéltek és írtak róla. Ma még azonban a probléma egzakt megfogalmazásáig sem jutottak el. Enélkül pedig a megoldáskeresés ön- és másamitásnál, illúziókergetésnél nem lehet több.

Mi az minőségügyi szempontból, ami a mai termelő és szolgáltató tevékenységben folyik? A legmagasabb színvonalat az ún. *minőségirányítás* képviseli. Ez nem azonos a hagyományos, ún. *minőségellenőrzéssel* (e kifejezés helytelen voltáról már szó volt), amely szintén minőségirányítás, de néhány bevált és egyszerű hagyományos módszer alkalmazását értik rajta.

A modern minőségirányítást *főleg az automatizált gyártás igényeinek kielégítésére fejlesztették ki.* Lényege nem elvi, hanem gyakorlati: nagymértékben automatizált. Ez a minőségirányítás a klasszikus irányításelmélet egy új (alkalmazási) fejezete. Elméletileg a minőséghez annyi köze van, hogy pl. nem hőmérsékletet vagy páratartalmat, hanem egy vagy több minőségjellemzőt igyekeznek irányítani. (E gyakorlati tudományágnak — az ún. *quality engineering*-nek — jó tankönyvei, folyóiratai vannak. Ismételjük: *ez a "tiszta" irányításelmélet egy gyakorlati alkalmazása a sok közül.* És mivel ez inkább specialisták területe, nem bocsátkozunk további részletekbe.)

Visszatérve a független gyárra, amelynek érvényesülése viselkedésétől függ, viselkedése pedig olyan jellemzőktől, amelyeknek csupán egyike másika minőségjellemző, kimondhatjuk azt, hogy az, ami igazán fontos, az a viselkedés minősége, ennek pedig a termékek minőségjellemzői *befolyásolói ugyan, de nem kizárólagos meghatározói.* A termék és szolgáltatás minősége a helyes viselkedés, a helyes navigáció kialakításában, illetve megvalósításában *akkor van a helyén, ha irányított szerepben van, alá van rendelve magának a viselkedésminőségnek, navigációminőségnek.*

A viselkedés minősége egzakt megközelítését két olyan feladattípus kapcsán kell elvégezni, amelyek különbözők, de a megoldásaikat tekintve vannak mindkettőnél egyaránt használható eljárások.

Az egyik feladat „abszolút" jellegű. Itt mindenkit (minden versenyzőt) egyénileg bírálnak el és hasznos, díja, értéke csak (tőle) magától, csak saját viselkedésétől függ. Ilyen az egyéni teljesítmény előre megállapított normák mellett. (Senki sem kap kevesebbet, mert a szomszédja többet „visz haza".) Egészen másképp ítéltetik meg a viselkedés minősége például csapatbajnokságnál. Nem jónak kell lenni, nem abszolút követelményeket kell teljesíteni, hanem minden résztvevőnél kell csak (az adott szempontokból) „jobbnak" lenni. Ilyen a háború is; elég az ellenséget legyőzni.

Befejezésül e problémák nehézséget szemléltető két érdekes jelenséget említünk.

Lehet egy csapaton belül valaki rossz, de így (is) hasznos és nélkülözhetetlen. Egy kocsma melletti étteremben alig hozott nyereséget, úgy hogy halaszthatatlan felújítását tulajdonosa nem tudta megoldani, ezért hirdetni kezdte, hogy eladja. A szomszéd — nem túl magas erkölcsű — kocsmáros megtudta ezt, és társulási szerződést ajánlott. *A társulás után sem ment az étterem sokkal jobban, de a közös haszon lényegesen magasabb lett.* A kocsmáros ugyanis előírta, hogy az étteremben annyira sósan kell főzni, amennyire csak lehet. Az agyonsózott ételek fogyasztása után a vendégek többsége igyekezett szomjat oltani — a legközelebbi — (a szomszédos) kocsmában.

Előfordul a labdarúgásban, hogy egyszerre ugranak fel fejelni játékosok és *összefejelnek.* Ezzel analóg jelenség a gazdasági konkurenciaharcban is előfordul. Több vállalat végez piackutatást. Nagy titokban tartják az eredményt. Kiderült ugyanis, hogy van valami, amire van kereslet, és a kínálat nem elégíti ki a keresletet. Mindegyik vállalat jó piackutatókkal rendelkezik. Mindegyik helyen ugyanarra az eredményre jutottak. Mindegyik vállalat nagy titokban, nagy erőkkel hozzálát, hogy a szőnbanforgó cikket mielőbb piacra dobhassa. Egyik vállalat sem kíméli a pénzt. És egymással egy időben mind meg is jelenik a piacon, hatalmas kínálatot (most már túlkínálatot) létrehozva. Az árak rövidesen esni kezdenek . . . És ott állnak a vállalatok az eladatlan áruval a tültetített piacon, mert még áron alul sem kell a portékájuk. Ilyen mintha nemcsak az iparban, nemcsak a mezőgazdaságban, hanem a számítástechnikában is előfordult volna már néhányszor . . .

Milyen tehát a jó minőségű viselkedés? *Más, ha abszolút és más, ha relatív az elbírálás. Más, ha egy független rendszerről van szó, és más, ha egymással kapcsolatban álló rendszerek rendszereiről vagy egy ilyen valamilyen részrendszeréről (pl. egy alkotó rendszerről) van szó.*

Milyen például egy labdarúgó viselkedésének minősége egy adott csapatban? Ugyanennek a viselkedésnek (működésnek) a minősége egy másik csapatban egészen más lehet.

Látható, hogy a problémák ízlelgetésénél nem jutottunk messzebb. *Lehetséges, hogy az ilyen problémák nem is tudományos értelemben vett egzakt és tudományosan megoldható problémák?* Elképzelhető.

A tudomány mindig törvényekkel dolgozik. Ezek mindig valamilyen megismételhetőséget feltételeznek. Az előzőekben vázolt problémák az élet igazi problémái. Az élet folyamataiban pedig nincs ugyanúgy megismételhetőség, van viszont újítás, újulás, vannak olyan jelenségek, amelyek azelőtt nem voltak és nem is lehet rájuk számítani. És ezek a nagy „előrevivő" tényezők kiszámíthatatlanok. Hogyan lennének tehát a szokványos tudományos eszközök alkalmazhatók?

Hogyan viselkedjen helyesen a minőségügyi szakember? Tárja fel a problémákat? Saját korlátait is? Vagy köntörfalazzon? Hítesse el a közvélemény által szentesített, de teljesen alaptalan dolgokat, mert másutt is ezt teszik? Mi az előbbi úton igyekeztünk járni.

Egy kis gondolkodnivalóval zárjuk fejtegetéseinket. Mindegyikünkben vannak vörös versejtek. Ezek úszkálnak a vérben. Működnek. Nem teljesen egyformán. Működésüknek tulajdonithatunk minőséget is. Egyik jobb minőségű viselkedést produkál, a másik kevésbé jobbat. Kinek a szempontjából? Az egyes vörös versejt szempontjából vagy az ember, „a tulajdonos" szempontjából? Lehetne erre azt mondani, hogy tessék egy kompromisszumos minőségjellemzőt alkotni, amelyben az ember és a vörös versejt érdekei arányosan szerepet kapnak. Igen ám, de az ember testének része a vörös versejt. Mennyiben vagyok én a vörös versejt? Ugye nem könnyű kérdések?

A számítástechnika nem ámítástechnika

Interjú az Alkotó Ifjúság vezetőivel

Az iparban, a mindennapi életben használható, értékesíthető megoldások megtalálása nem kizárólag az idős szakemberek privilégiuma! Ma már egyre inkább terjed az a felismerés, hogy esetleg még egy általános vagy éppenséggel egy középiskolás fiatal is képes egyes területeken piacképes ötletek létrehozására, csak komolyan kell foglalkozni az általuk felvetett gondolattal. Ez a felvetés kissé különösnek hangzott hazánkban öt esztendővel ezelőtt, amikor a fiatalok és különösképpen a fiatal szakemberek ötleteinek felkarolására a KISZ KB az Állami Fejlesztési Bankkal közösen meghirdette az „Alkotó Ifjúság"-pályázatot, majd ennek menedzselésére, a beérkezett pályaművek hasznosítására létre hozták az Alkotó Ifjúság Egyesülést, amely napjainkra már jócskán túlnőtte egykori kereteit. Most már dinamikusan fejlődő és változó szolgáltató- menedzser-fejlesztő-innovatív formában tevékenykedik. Budapesti központjában négy szakirodában, az észak-magyarországi területi képviseletben integrált formában folyik ez a komplex munka, amelyet kiegészít a tíz megyei képviseleten folyó piaci feltáró és komplex fejlesztő tevékenység. Több száz újítás, szabadalom, üzleti sikerek és egypár kevésbé eredményes próbálkozás jellemzi tevékenységüket, amelyet a számítástechnika korai időszakban történő megjelenése, majd napjainkig tartó egyre dinamikusabb felfutása fémjelez. E szerteágazó tevékenységről beszélgettünk Kulcsár Istvánnal, az Alkotó Ifjúság Egyesülés budapesti számítástechnikai irodájának vezetőjével, valamint dr. Csomai Zoltánnal, aki az észak-magyarországi területi képviseletet vezeti.

µM: *Egy olyan menedzser jellegű vállalat, mint az Alkotó Ifjúság Egyesülés hogyan válhatott a számítástechnika területén szolgáltató vállalattá?*

K. I.: Alig egy évvel az egyesülés beindulása után nyilvánvalóvá vált, hogy nem nélkülözheti a számítástechnikát. Sem saját belső munkájában, sem abban, hogy eladásra alkalmazói szoftvereket menedzseljen. Ehhez viszont kellett egy gép is. Akkoriban a PC-k nem voltak még elterjedve, így csakis nagyszámítógép jöhetett szóba. Az OMFB-nél megpályáztunk egy ilyen irányú támogatást, amire 20 millió forintot kaptunk. Ehhez viszont az a feltétel csatlakozott, hogy gépünkkel meg kell oldani az Országos Találmányi Hivatal számítástechnikai feladatait. Így sikerült egy PDP kompatibilis magyar TPA gépet beállítani, amely ma is üzemel. Elsősorban a találmányi bejelentésekkel kapcsolatos adminisztrációt intézi az OTH részére. Működik már a megadott magyar szabadalmak nyilvántartása, fejlesztés alatt áll a védjegyregiszter. Emellett az OTH nyilvántartásait és statisztikáit is e gép programrendszereivel készítik. A fennmaradó időben a gépet az egyesülés feladataira lehet felhasználni. Természetes, hogy az első időszakban forgalmazott szoftverek is e géptípusra íródtak. Mintegy 15 TPA rendszerszoftver-kiegészítőt forgalmaztunk belföldre és a SZÁMALK—OSAK megállapodás keretében a Szovjetunióba is. Ekkor kezdődött a komplex számítógépes szolgáltatás is: a TPA-t készítő KFKI-vel közösen alkalmazói és rendszerszoftver-tanfolyamokat és dokumentációkat hoztunk létre.

Cs. Z.: Miskolcon a számítógépes munka később, a PC-k elterjedésével kezdődött, 1984-ben. Ennek oka egyszerű volt: nem volt nagygépünk. Ezután az egyesülés kínálata ugrásszerűen megnőtt. Jelenleg több mint 128-féle szoftver van forgalomban, természetesen a TPA-SzM 4 nagygépre valamint a Proper, MO8X, Apple, IBM PC és a Commodore sorozat gépeire. Még mindig a legszélesebb a kínálat Commodore szoftverekből, a több mint 90 program között ügyviteli, matematikai, tervezési és mezőgazdasági feladatokat megoldó programcsomagot találhat a vásárló. Ezen túlmenően, az egyesülés keretén belül a hardverújdonságokat is fejlesztjük és menedzseljük. A MOM megvásárolta a floppy- és winchesterillesztő egységünket, amelyet jövő évtől várhatóan belföldre és szocialista relációba nagy sorozatban fog készíteni.

µM: *Sok hazai szoftver-előállító céget ért kimondottan vagy éppen kimondatlanul a vád: A „szerűség" terjesztői. Hozzájárulnak ahhoz, hogy a számítástechnika „ámítástechnika" legyen, azaz segítik a vállalatokat abban, hogy kicsiny kapacitású személyi számítógépekre, például C—64-re vagy Sinclair Spectrumra építhessék számítástechnikai rendszerüket. Már a kezdet kezdetén nyilvánvaló, hogy ezek a szinte játékszámba menő — bár olcsó — gépek nem alkalmasak komoly feladatok ellátására. Így a vállalat kipipálja a megoldott feladatot, de az megmarad, sőt a helyzet súlyosbodik . . .*

Cs. Z.: A probléma valójában létezik, de . . . Ez viszont a vállalatokon múlik. Ugyanis, amikor felkérnek bennünket egy probléma megoldására, megvizsgáljuk a cég adottságait, s több változatban megoldási javaslatot teszünk. Általában egy olcsó — szükségmegoldásnak is felfogható —, egy közepes és egy a hosszú távú igényeket is figyelembe vevő változatot. Mi a javaslatainkban legalább a közepes kategóriájú PC-k beszerzését javasoljuk, pl. IBM vagy APPLE, esetleg IBM kompatíbilis más gépen alapuló rendszert. Ilyenkor a válasz ismert: Sajnos nincs pénz . . . Így maradnak a C—64-ek vagy valami hasonló megoldás. Ilyenkor azonban nem szabad csodálkozni, hogy a vállalatnál rövidesen eljutnak a kérdésben felvetett problémához. S az egyesülés is csak azt szállíthatja, amire megrendelés és fizetőképes kereslet van . . .

K. I.: Ennek ellenére 1986—1987-ben szeretnénk teljes egészében áttérni a gazdasági célú programoknál, de az ipari és a mezőgazdasági egyéb szoftvereknél is az IBM PC és azzal kompatibilis gépek alkalmazására. Így már a javaslatban sem szerepel egy idő után Commodore 64 mint alternatíva. Bár feltétlen érdemes megjegyezni, hogy a kicsiny, mondhatni játékgépeken futtatott komoly felhasználói programok elkészítése csak igen fejlett programozói ismeretekkel, nagy rutinnal és gyakorlattal eredményes. Ugyanis egy olyan program, amely simán és gyorsan lefut egy IBM PC XT-n, nem fér bele egy C—64-be. Ilyenkor rafinált módon szét kell tördelni a programot, s olyan fogásokat kell alkalmazni, hogy a folytonos háttértárhasználat mellett a program futása elviselhető ideig tartson, ne legyen lassú. Ha egy ilyen „turbó" programot összehasonlítunk a megfelelő gépen futtatottal, annál alig lassúbb. Ez a programozói tudást dicséri. Sőt alkalmat adhat a nagyobb gépek ilyesfajta „megpatkolására". Ez feltétlen előreviszi a programozási technika fejlődését.

Cs. Z.: A sokat emlegetett „szerűség" tehát a pillanatnyi kisebb beruházási igény következménye. Ugyanis jelenleg senkit sem érdekel, hogy egy vagy két esztendő múlva esetleg nagyságrendekkel nagyobb összeget kell majd beruházni, mint most egy drága, de hosszú távra előrelátó beruházás igénye lett volna. Emellett a szemlélettel is baj van. Legtöbb helyen elzárkóznak a számítástechnika alkalmazásától. Ha a középkorban lennénk, akkor biztosan az „ördög művének" kiáltanák ki sok helyen. Rébusznak veszik a gép használatát, s nem is akarják megtanulni az alkalmazást. A kitartás azonban itt is sikerre vezethet. Ilyen esetünk volt az egyik észak-magyarországi vállalattal. Amikor először felkeresték őket, finoman, de eltanácsoltak bennünket. Második alkalommal „véletlenül" egy Commodore 64-gyel és egypár játékprogrammal jártunk arra. Mivel az egyik ottani vezetőt érdekelte, hogyan játszik egy bizonyos, általa kedvelt kártyajátékot a gép, hajlandó volt kipróbálni. Mondanom sem kell, megszerette s rövid időn belül egy szerény mértékű vállalati megrendelést tudhattunk a magunkénak.

µM: *Főleg milyen korúak ötleteivel foglalkoznak?*

Cs. Z.: Nálunk is megfigyelhető az a tendencia, amelyről olvashatunk a világsajtóban. Egyre fiatalabbak azok, akik a szoftverötleteikkel megkeresnek bennünket. Egyik legfiatalabb szerzőnk 12 éves(!), s sok segédprogramot készített el őreg programozókat megszégyenítő ügyességgel édesapja programötleteihez. Éppen ezért nagyon nagy tartalékok vannak az országban. S ezért kérjük azokat, akik úgy érzik, hasznosítható szoftver- vagy akár hardverötletük, találmányuk van, keressenek fel bennünket. (Cím: Alkotó Ifjúság Egyesülés, Bp. V., Garibaldi u. 2 Számítás-

technikai iroda. Tel.: 112-666 vagy Miskolc, Győri kapu 21. Tel.: 85-010.) Itt megvizsgáljuk ötleteiket, és ha értékesíthető, menedzseljük. S csak az ötlet jósága a mérvadó, korhatár, képzettség, szakma nem számít!

K. T.: Az egyesülés jelenleg még nem rendelkezik külkereskedelmi joggal, de ha szükségesnek látszik, akkor kérvényezni fogjuk. Gazdasági rendszerünk annyira eltér a nyugatitól, hogy ilyen témájú termékeinket nem tudjuk ott értékesíteni. Ennek ellenére felmerült a gondolat a rendelkezésre álló hatalmas szellemi bázis jobb kihasználására, például bérmunka keretében. De a kilépés a világpiacra küszöbön áll. Például az észak-magyarországi területi képviselet szoros kapcsolatokat épített ki az NSZK-beli Kornhoffer céggel, amely a Multitec rendszert képviseli. Az így szerzett tudás is növeli a munka színvonalát.

µM: *S a gyakran változó gazdasági környezet, a dinamikusan átalakuló pénzügyi szabályozórendszer mennyiben befolyásolja terveiket? Korábban prosperáló kisvállalkozások — a számítástechnikában van belőlük egynéhány — mennek csődbe, változnak az adók, nőnek az elvonások.*

Cs. I.: Mi az északi régióban a nyitással a kisvállalkozások felé megpróbáljuk megakadályozni, hogy az ott kialakult szellemi tőke, gyártási tapasztalat veszendőbe menjen a gazdaság számára. Tudomásul vesszük azt, hogy a szabályozóváltozások tendenciái a magán kisvállalkozások részleges állami — kisszövetkezeti — keretekbe történő terelésére irányulnak. A miskolci tanácscsal az Alkotó Ifjúság Egyesülés kötött egy olyan együttműködési megállapodást, hogy számos kihasználatlan ipari és egyéb épületet megkaphatunk, ahol elhelyezhetjük a velünk együttműködő kisvállalkozásokat akár gazdasági társaság, akár más formában történő együttműködés keretében. Így kialakulhatnak az éltechnológia ottani bázisai is. Akik túlélték a szabályozás változásainak viharait, azoknál feltétlen egy olyan innovatív képesség kapcsolódott a vállalkozókészséggel és szakmai tudással, ami létfontosságú.

K. I.: Ami viszont az ösztözőket illeti, ott egy kedvezőtlen változás bőszítette fel a szoftverek szerzőit. Egy új rendelet alapján a Szerzői Jogvédő Hivatalon keresztül kifizetett jogdíjat ezután 40% TBK- és 10% béradó terheli. Ezért, hogy rentábilis legyen a forgalmazásunk, csökkenteni kell a szerzők díjazását, módosítani a szerződést. Csak az a meggondolandó ezzel a szabályozással, nem lött-e öngólt e rendelet megalkotója: a költségvetési bevétel növekményénél nem jóval nagyobb-e az a kár, amelyet a nem hasznosított szoftverek okoznak?

µM: *Viszonylag kevés szó esett az elektronikáról, a hardverről. Azok innoválásában mennyire képesek részt venni?*

Cs. Z.: Mint említettem, a városi tanácstól kapott épületekben szeretnénk megkezdeni egy nagyobb mérvű fejlesztéssel összekapcsolt gyártást. Elsősorban az alapgépek bizonyos kiegészítőit, például illesztőegységeket akarunk készíteni. De már az egyesülés most is mutathat fel e területen is sikereket. Ilyen, nagy érdeklődésre számot tartó produktumunk a COMES illesztőegység, amely a Commodore és más e kategóriába tartozó számítógép és az IBM, illetve ESZR rendszerek nagygépei közötti kommunikációt biztosítja. Vagy a másik, a négyterminálos multiplexer, amely egy bérelt vonal és egy modem használatával négy terminál számára szükséges adatátviteli feladatokat képes ellátni. Ez a mostani vonalinséges időkben szinte megfizethetetlen előny. Így alakult ki a jó kapcsolat is a Postával, hiszen ők nemcsak szoftverjeinket használják, hanem e termékünkkel bizonyos felhasználói nyomást is leveszünk a vállukról. Így nem csoda, hogy a multiplexert ők is ajánlják a felhasználóknak.

µM: *Nem érzik az egyre szaporodó számítástechnikai kis- és nagyvállalkozások okozta konkurenciát?*

K. I.: Megpróbálunk mindennel foglalkozni, amivel érdemes. Hazánkban szerencsére elég széles a versenypálya, egymás mellett van még elegendő hely a piacon. Bár vannak azonos tevékenységi formáink, mégsem vagyunk a szó hagyományos értelmében konkurensek. Egyesítjük az általában „tisztán" előforduló vállalati funkciókat: egyaránt vagyunk műszaki fejlesztővállalat éppúgy, mint innovációs szakcég, lizingbeadó és szakértő-szervező szakvállalat. Ugyanakkor a kis egységekből álló felépítés, a fiatal szakembergárda nagyfokú rugalmasságra ad módot. S csak a produktumot néző szemléletünkkel a meritési bázisunk is nagyobb, mert hozzánk be mer jönni egy olyan fiatal szakmunkás, szakember, esetleg tanuló, aki nem tenné be a lábát például egy mammutvállalathoz, hogy kilincseljen ötletével. A fiatalok vállalata vagyunk, s megpróbálunk ekképp cselekedni.

µM: *Miként kapcsolódnak be az oktatásba, a számítástechnikai ismeretterjesztésbe?*

K. I.: Mi szereltük fel a Szabolcs Szatmár Megyei KISZ-bizottság számítástechnikai klubját, valamint a politikai képzési központot is. Az természetes — hiszen van némi közünk a fiatalokhoz és a KISZ-hez! — hogyha segítséget kérnek tőlünk, megpróbálunk tenni valamit, például a hardver áráért segítjük a KISZ számítástechnikai klubjait. Az NJSZT és a KISZ Heves megyei számítástechnikai napján éppúgy ott voltunk, mint az augusztusi miskolci helyi ipari kiállításon és vásáron észak-magyarországi területi képviseletünk révén. Ilyenkor nemcsak a szakembereket, hanem az érdeklődőket is várjuk, mert szeretnénk megismertetni és megkedveltetni a számítástechnikát.

Cs. Z.: A múlt esztendő végén szűkebb pátriánkban, azaz itt Miskolcon 20 általános iskolának telepítettünk megrendelésükre Commodore 16-os gépet és a szükséges perifériákat, gondoskodva a szükséges betanítási és más feladatokról is. Ezenkívül keressük a lehetőséget, hogyan tudnánk részt venni tevékenyen tantárgyi és egyéb oktatóprogramok írásában. Egyelőre itt még kétségek merülnek fel: az egyes tantárgyakhoz, például földrajzhoz, nyelvhez, biológiához, kémiához, milyen típusú programokat kell íratni. Ugyanis oktatási anyagunk, iskolarendszerünk más országoktól eltérő lévén, nem lehet azok tapasztalatait adaptálni. Nekünk is végig kell járni a megismerés útjait. Az iparban ez a tevékenységünk már régebben folyik, mert tanítjuk a gépek kezelését, s bár nem adhatunk bizonyítványokat, mégis sikerrel és nagy érdeklődéssel tartjuk tanfolyamainkat. Ez a továbblépés útja is e részterületen: bekapcsolódni a számítógép alkalmazását oktató tanfolyami rendszerbe, programba.

KIS JÁNOS

A játékprogramozás technikája

A játék irányítása

A *36. listában* látható rész a helikopter sebességét szabályozza. Ha a botkormány tűzelőgombját lenyomjuk, a 6528Ø memóriacímen a legalsó bit értéke Ø lesz. Ezt észlelve a program a sebesség (SP) egyébként 3 egység értékét 8-ra növeli („turbó"-érték). Gyors mozgás esetén az üzemanyag is gyorsan fogy (FU csökkenése 1 egység, egyébként Ø,75), és a helikopter útján törlendő terület is gyorsabban változik.

A *37. listában* a botkormány állapotát vizsgáló rész látható. A Dragon, az Apple, az IBM PC gépek ún. analóg botkormányt használnak. Ezeknél a bot irányával két egymásra merőlegesen elhelyezett potenciométer skáláján mozgatjuk a tolókát. Ez tehát két változó ellenállásértéket ad. A két, X, illetve Y irányú elmozdulással többé-kevésbé arányos ellenállásérték a számítógép számára szolgáltatott analóg jelpár. A Dragon analóg bemenetei 6 bit felbontásúak, vagyis 1:63 tartományt fognak át. A program ezt nem használja ki, csak (az ún. Atari típusú kapcsolós botkormányoknál, amilyet a Commodore gépek is használnak) a 8 fő irányba tett elmozdulásokat vizsgálja. Így az X irány (J):

— balra $J<3$,
— jobbra $J>6Ø$,
— nincs elmozdulás $3 \leq J \leq 6Ø$.

Az Y irány (J1) hasonló. A botkormány a JOYSTK—BASIC utasítással olvasható be.

A 8ØØ-as sorban azt vizsgáljuk, hogy földet ért-e a helikopter. Amennyiben nem, úgy a lista hátralevő részét és a 38. listában szereplő részt átugorja. A lista maradékában azt vizsgáljuk, hogy a földet érés jó helyen történne-e meg (itt $94<X<1Ø6$ értékű)? Itt csak függőleges mozgás lehetséges.

A *38. listában* a földet érés miatt újrarajzoljuk a repülőtér négyszögét.

A *39. listában* a rajzteret korlátozzuk, és megakadályozzuk, hogy ezt a területet a helikopter elhagyja.

Az utolsó, még nem vizsgált ténykedésünk a repülőtéren földet ért helikopter üzemanyaggal történő feltöltése, növekvő magassággú hang adása mellett. A hang jelzi a feltöltöttséget. Túltölteni nem lehet, mert az üzemanyag mennyiségét a 92Ø-as sorban korlátoztuk. Ez a *40. listában* történik meg.

Összefoglalás

A bemutatott program fő előnye, hogy bár minden játékprogram-jellemzőt tartalmazott, mégis elkerülte a gépfüggetlen tárgyalásmódot lehetetlenné tevő gépi kódú programozást; még gépi kódú részleteket sem tartalmazott. A használt BASIC-változatban szerepeltek ugyan speciális utasítások is, amelyek azonban alig térnek el az IBM PC és az MSX gépek hasonló célú utasításaitól, de az adott magyarázatok itt is segítettek.

Ez a BASIC program ismertetése nem azt a célt szolgálta, hogy ezután bárki, a programot a saját gépére átírva, ugyanezt a programot a saját gépén BASIC programként használhassa, mert ezt a legtöbb gépen a grafikus utasítások végrehajtási sebessége lehetetlenné teszi. A program ilyen részletességű ismertetése tehát nem egy „elfogyasztható étel" előállítását szolgálja, még csak azt sem, hogy ilyenhez receptet adjon — kivéve az említett gépeknél —, hanem annak a módnak a bemutatását, ahogy egy ilyen recept elkészíthető.

GALINA FERENC

```
730 F=PEEK(65280):IFF=126ORF=254THEN
SP=8:LINE(X-3,Y-2)-(X+23,Y+15),
PRESET,BF:FU=FU-1ELSESP=3
```

36. lista

```
740 J=JOYSTK(0):J1=JOYSTK(1)
750 IFJ>60THENX=X+SP:DI=1
760 IFJ<3THENX=X-SP:DI=2
770 IFJ>3ANDJ<60THENDI=3
780 IFJ1>60THENY=Y+SP
790 IFJ1<3THENY=Y-SP
800 IFY<125THEN870
810 IFX<95ANDJ1>60THENY=Y-SP
820 IFX<95ANDJ<3THENX=X+SP
830 IFX>105ANDJ1>60THENY=Y-SP
840 IFX>105ANDJ>60THENX=X-SP
```

37. lista

```
850 LINE(90,160)-(130,160),PSET
860 DRAW"BM94,160;G10R55H10"
```

38. lista

```
870 IFY<25THENY=25
880 IFX<5THENX=5
890 IFX>227THENX=227
```

39. lista

40. lista

```
900 IFY>155THENY=155:FU=FU+8:SOUNDFU,1
910 GOTO990
```

SPECTRUM

Black Jack

A játék magyar megfelelője a magyar kártyával játszott huszonegyezés. Egyszerre ketten játszhatnak: az egyik játékos a gép. Induláskor először tétet kell tenni, amelynek maximális értéke 1000 lehet. Mindketten húznak a kártyapakliból. A kártyák összegének 21-nek kell lennie, de a partner kártyáinak összegénél lehetőleg nagyobbnak.

A gép húz és eldönti, hogy elég-e neki a lapok összértéke (500—580-as sorok).

9020 PRINT Ha a lapok összege 17-nél több, akkor kiírja az összeget (jobb felső sarok), a lapoknak megfelelő értéket (bal felső sarok). Ezután a játékos eldöntheti, hogy kér-e még lapot vagy sem. A játékos az l gombbal kérhet lapot, az n gombbal jelzi, hogy nem kér. Ha az n gombot megnyomjuk, vagy a kért lapok összege nagyobb 21-nél, akkor a gép kiírja a nyertes nevét és a nyerési arányt.

9030 PRINT Döntetlen esetén nem változik az érték. Az a megnyomására a játék befejeződik. A játékot addig játszhatjuk, amíg a játékos pontjainak (pénzének) összege 0-ra nem csökken. Ezután újra kezdhetjük a játékot.

9040 PRINT Főbb változók: a tét összege = a (kezdő a=1000)
a tét=al
a kártyák összege=p (később p=q)
a gép összege=q
a játékos összege=k
kártyahúzás=k

A 6020—6030 sorok elhagyhatók, csak a kazettára mentést segítik.

FORGÁCS BALÁZS

```
  1 CLS : PAUSE 20: LET j=0
  3 LET f=1
  5 LET p=0
  6 LET q=0
 10 LET a=1000: LET j=j+1
 15 PRINT "  ■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■"
 17 PRINT "
 "
 20 PRINT "  ))))))))))"; INK 2;
"BLACK JACK"; INK 0;"(((((((((("
 25 PRINT "
 "
 27 PRINT "  ■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■"
 28 IF j=1 THEN GO SUB 800: PAU
SE 50
 30 IF a=0 THEN PRINT "A tele
d ";a;" lehet. Új játékot    ke
ll kezdened !!!                1-
STOP                           2-
Új játék ": GO SUB 1100
 35 BORDER 7
 40 PRINT AT 19,0;"A tét maximu
m:";a;" lehet,ebből    gazdátkod
j"
 50 INPUT "Mennyi a tét? ";a1
 60 IF a1<0 OR a1>a THEN GO TO
40
 70 PRINT "Nyomd meg az &-t a &
tárlhoz!"
 75 PAUSE 0
 76 PRINT
 80 LET p=0
 90 LET k=INT (RND*7)+1
100 GO SUB 440
110 IF p>21 THEN PRINT "    Te n
yertél": LET a=a+1: GO TO 700
115 LET q=p
120 IF p<=16 THEN GO TO 90
130 IF p>=16 AND p<=21 THEN PRI
NT "Te jössz"          a gép ";
p;" rakott!"
 490 LET q=p: GO TO 600
 499 PRINT
 500 IF k=1 THEN PRINT " asz";
LET p=p+11
 510 IF k=2 THEN PRINT " király"
: LET p=p+5
 520 IF k=3 THEN PRINT " felso";
: LET p=p+4
 530 IF k=4 THEN PRINT " alsó";
LET p=p+3
 540 IF k=5 THEN PRINT " VII";:
LET p=p+7
 550 IF k=6 THEN PRINT "VIII";"
LET p=p+8
 560 IF k=7 THEN PRINT " IX";: L
ET p=p+9
 570 IF k=8 THEN PRINT " X";: LE
T p=p+10
 580 RETURN
 600: PAUSE 50: CLS : GO SUB 604
 603 GO SUB 620
 604 PRINT AT 2,1;"  A GOMBOK:";
AT 2,0;"l-még egy lap";AT 3,0;"n
-elég";AT 4,0;"&- STOP"
 610 PRINT INK 3;AT 2,15;"A gép
lapjainak";AT 3,15;"az összege:"
;q
 615 GO SUB 1200
 617 RETURN
 620 REM a játékos kártyái
 625 PRINT INK 7;"      A játékos
kártyái:"
 626 LET p=0
 627 PRINT
 630 LET k=INT (RND*7)+1: GO SUB
500
 634 LET w=2
 640 IF p>21 THEN PRINT TAB 15;"
En nyertem": LET a=a-a1: GO TO 7
00
 650 PRINT TAB 15;" kérsz még?"
 655 PAUSE 0
 660 IF INKEY$="l" THEN GO TO 63
0
 670 IF INKEY$="n" THEN GO TO 90
0
 675 IF INKEY$="a" THEN STOP
 680 GO TO 655
 700 PRINT : PRINT INK 2;"NYOMD
MEG VALAMELYIK GOMBOT A    FOL
YTATÁSHOZ !": PRINT : PAUSE 0:
CLS : GO TO 15
 750 STOP
 800 PRINT "  INK 1;" Készített
e:  INK 0;" FORGÁCS BALÁZS @"
;"1985.IV.19. BP ----O-----"
 810 PAUSE 30: PRINT "A lapok
összegének kiírója 21-nek kell le
nnie,ma tobb, akkor automat
ikusan a másik nyer,ha    tévesebb
,akkor az nyer, akinek    a kárty
ák összértéke a több."
 820 PRINT "-A lapok értéke:  "
;"also=3,felso=4,király=5,ász=11"
;" a többi saját értéket képvisel
i."
 830 PAUSE 400: FOR j=11 TO 19:
PAUSE 5: PRINT AT j,0,"
 "
 840 NEXT j
 890 RETURN
 900 PRINT "  IF q=p THEN PRIN
T "))) Döntetlen ((( ": a=a: "
";p
 910 IF q>p THEN PRINT " ))) en
nyertem   ((( ";q;"-";p: LET a
=a-a1
 920 IF q<p THEN PRINT " ((( Te
nyertél ((( ";p;"-";q: LET a=
a+a1
 930 PAUSE 200: CLS : GO TO 15
1100 PAUSE 0: IF INKEY$="1" THEN
STOP
1110 IF INKEY$="2" THEN RUN
1199 STOP
1200 PRINT AT 5,0; INK 2;"Erteke
k:  INK 1;AT 5,0;"also=3  felso
=4  király=5    ász=11"; PRIN
T  INK 0;"a többi saját értéku"
1210 RETURN
6020 SAVE "black jack" LINE 1
6025 PAUSE 100
6030 VERIFY "black jack"
```

A szerző szerint a programozás „konstruktív tudomány". Hibás tehát az az elképzelés, mely szerint az ismeretek átadása előbb-utóbb valamiféle „receptgyűjtemény" betanításává egyszerűsíthető. *A jó példák tanulmányozása azonban igen hasznos lehet.* A könyvben bemutatott programok egyúttal jó *programolvasási gyakorlatul* is szolgálnak. A szerző az olvasót bevonja a programok fokozatos kialakításába, különböző „pillanatfelvételeket" mutat be egy-egy program fejlődéséről, a lépésenkénti finomítás módszeréről. A bemutatott programok számítógépen valóban le is futnak.

Itt most figyelemfelkeltőül a könyv „Rendezések" című fejezetéből mutatunk be tanulságos részleteket. A rendezés (sorting) a dolgok egy meghatározott halmazának valamilyen sorrend szerinti átrendezését jelenti. A rendezés feladata az, hogy megkönnyítse az elemek későbbi keresését. A rendezés kiemelkedően fontos tevékenység — különösen az adatfeldolgozásban —, így különösen ideális téma az algoritmusok sokféleségének bemutatására: például ugyanazt a feladatot végző algoritmusok közül több bizonyos értelemben optimális és a legtöbb algoritmusnak van valamilyen előnye a többivel szemben. Például az egyik csak kis mértékben rendezetlen tömböknél, a másik a teljesen rendezetleneknél ideális.

A rendezőmódszereket általában a következő két kategóriába szokás sorolni:

— tömbök rendezése és
— (soros) fájlok rendezése.

E két kategóriát gyakran nevezik *belső*, illetve *külső* rendezésnek is, utalva arra, hogy a tömbök rendezése a számítógép gyors, véletlen (random) elérésű, „belső" tárában zajlik, amíg a fájlokat lassúbb, de tágasabb „külső" tárra szokás elhelyezni, melyek mechanikusan mozgatható berendezések (lemezek és szalagok). A kétfajta módszer különböző, az alábbiakban csak a tömbök rendezéséből adunk ízelítőt.

A könyvben bemutatott rendező algoritmusok olyanok, amelyek a tételek átrendezését saját helyükön hajtják végre: tehát a rendezéshez nem vesznek igénybe újabb tárterületet lefoglaló segédtömböt vagy tömböket. Az algoritmusok eltérnek egymástól például a rendezéshez szükséges összehasonlítások és cserék számában. Minél kevesebb összehasonlítással és cserével jut el az algoritmus a teljes rendezésig, annál jobbnak minősíthető. Először egy ún. közvetlen módszert mutatunk be röviden, jelzésszerűen. Ez a módszer nem a legjobb, de rövid, egyszerű és alkalmas a rendező elvek jellemzőinek megvilágítására. Az itt bemutatott rendezés közvetlen beszúrással történik. A rendezendő tételeket elvben két részre osztjuk:

— az $a_1, \ldots, a_{i-1}$ fogadó sorozatra és
— az $a_i, \ldots, a_n$ *leadó* sorozatra.

Minden lépésben ‚i=2-től egyesével n-ig, kiemeljük a leadó sorozat első elemét és áttesszük a fogadó sorozatba, *beszúrva* őt a megfelelő helyre. Először a tömböt a második tételnél választom ketté, először a második elemet emelem ki és *szúrom* be a most még csak egyetlen elemet (az elsőt) tartalmazó *fogadó* „sorozatba".

Az első lépés után a_1 és a_2 sorrendje már helyes lesz. Ezután lépek egyet jobbra és a harmadik elemnél választom ketté a tömböt fogadó és leadó részre. Kiemelem a harmadik elemet és beszúrom a már rendezett a_1 a_2 sorozatba. És így tovább:

```
for i:=2 to n do
    begin
        x:=a[i]; {„kiemelem" a soron következő
                  tételt}
```

ide jön az a rész, amelyik x-et beszúrja az $a_1, \ldots, a_{i-1}$ fogadó sorozat megfelelő helyére

```
    end.
```

A bekeretezett részt, a *beszúrást* úgy valósítjuk meg, hogy a kiemelt tételt „visszafelé", a fogadó sorozat végétől az eleje felé haladva sorban összehasonlítjuk a fogadó sorozat egyes tagjaival egészen addig, amíg x helyét meg nem találjuk. Az alábbi programban vegyük észre, hogy *két* különböző feltétel is van, melyre ez a visszafelé haladó „rostáló" eljárás befejeződhet:

1. Találunk egy a, tételt, amely x-nél kisebb.
2. Elértünk a fogadó sorozat bal szélére (mert a kiemelt tétel helye a fogadó sorozat elején van).

A második esetet a program az ún. „strázsa" (őrszem, sentinel)-technikával kezeli. Az alábbi program éppen a kiemelt x:=a[i] elemet állítja „őrszemként" a sorozat elejére (a[0]:=x). Emiatt a rendezéshez igénybe vett tömb egy tétellel hosszabb, mint a rendezendő sorozat (éppen az a [0]-val).

Tehát a rendezés közvetlen beszúrással:

```
procedure közvetlen beszúrás;
        type tétel=  record kulcs: integer;
                     {egyéb komponensek}
                     end;
             index=  0 . . . n; {Figyelem, a 0 a strá-
                     zsa helye!}
    var i,j: index;
        x: tétel;
        a: array [index] of tétel;

begin for i:=2 to n do {a fogadó és leadó soro-
                        zat határa}
    begin x:= a[i]; {a kiemelés}
          a [0]:=x; {a „strázsa" felállítása}
          j:= i-1;
          while x. kulcs <a[j]. kulcs do
          begin a[j+1]:=a[j];
                j:=j-1; {haladunk visszafelé a fogadó sorozatban}
          end;
          a[j+1] :=x; {itt a beszúrás}
    end
end
```

A könyv *elemzi* is a közvetlen beszúró algoritmust: megállapítja, hogy

— az i-edik „rostálás" során a kulcs-összehasonlítások C_i száma legalább 1 és legfeljebb i−1, tehát némi egyszerűsítéssel átlagosan i/2.
— A mozgatások (tehát a tételek tárolásának) száma:
$M_i = C_i + 2$ (a strázsát is beszámítva).
— Az összehasonlítások és a mozgatások teljes száma így:

$$C_{min} = n-1, \qquad M_{min} = 2(n-1),$$

$$C_{átl} = \frac{1}{4}(n^2+n-2), \qquad M_{átl} = \frac{1}{4}(n^2+9n-10),$$

$$C_{max} = \frac{1}{2}(n^2+n)-1, \quad M_{max} = \frac{1}{2}(n^2+3n-4).$$

A legkisebb értékek akkor adódnak, ha a tételek eredetileg sorrendben voltak; a legrosszabb eset (a „worst case") az, ha a tételek eredetileg fordított sorrendben álltak. Ebben az értelemben a fenti algoritmus *viselkedése természetesnek* mondható. Látható, hogy a megadott algoritmus egyúttal *stabil* rendezőeljárást ír le: az azonos kulcsú tételek eredeti sorrendje nem változik.

A közvetlen beszúró algoritmus — úgy tűnik — egyszerűen javítható, ha észrevesszük, hogy az $a_1, \ldots, a_{i-1}$ fogadó sorozat, ahova az új tételt beszúrjuk, már rendezett. Kézenfekvő a „bináris" keresés alkalmazása, melyben a sorozat felezését addig ismételjük, amíg a beszúrás helyéhez el nem érkezünk:

```
procedure bináris beszúrás;
        type: {mint az előző algoritmusnál}
        var i,j,l,r,m:index;
            x:tétel;
            a:array [index] of tétel;
begin
        for i:=2 to n do
        begin x:=a[i]; l:=1; r:=i-1;
              while l ≤ r do
              begin m:= (l+r) div 2
                    if x. kulcs <a[m].kulcs
                    then
                         r:=m-1        else
                         l:=m+1
        end;
        for j:=i-1 downto l do a[j+1] :=a[j];
        a[l] :=x;
        end
end
```

A bináris beszúrás elemzése (amit itt elhagytunk) azt mutatja, hogy az elért javítás csak az összehasonlítások számára vonatkozik, a szükséges mozgatásokra azonban nem, így nem érhető el ugrásszerű javulás. Nem is beszélve arról, hogy egy rendezett tömb újrarendezése ezzel több időt venne igénybe, mint közvetlen beszúrással. Ez a példa jól mutatja, hogy egy „kézenfekvő javításnak" gyakran kisebb jelentőségűek a következményei, mint amit az ember első pillantásra remél, sőt néhány esetben (elő is fordul) a „javítás" valójában rontást eredményez.

A közvetlen beszúró rendezés továbbfejlesztését D. L. Shell vetette fel 1959-ben. A *Shell-rendező* úgy működik, hogy először egymástól távolabb álló elemeket hasonlít össze és cserél ki, miután így végigpásztázta a tömböt, a távolságot csökkenti a két elem között és megismétli az összehasonlításokat és (szükség szerint) a cseréket. Az utolsó menetben szomszédos elemeket hasonlít össze és cserél ki. A csökkenő távolságokat („a fogyó növekményeket") különböző módon meghatározhatjuk. Az alábbi program olyan, hogy a t darab növekményt („távolságot") a h tömbben helyezhetjük el. Ahhoz, hogy az algoritmus működjék, ki kell elégíteni az alábbi követelményeket:

$$h[t]=1 \text{ és } h_{i+1} < h_i$$

Természetesen minden egyes menetnek saját strázsát kell állítania. Ezért a tömböt nemcsak

évekkel ezelőtt egy másik (D. Miller könyvéről az előző számban adtunk tudósítást), ma is nagyon aktuális, jó számítástechnikai „olvasókönyvet", mely nemrég magyarul is megjelent. (N. Wirth: Algoritmusok + Adatstruktúrák = Programok; Műszaki Könyvkiadó 1982, fordította: Lehel Jenő okl. matematikus, lektorálta: Bószörményi László, okl. villamosmérnök). A magyar fordítás az üzletekben rég elfogyott, a könyv hiánycikk a könyvtárakban is, az angol nyelvű kiadás (N. Wirth: Algorithms + Data Structures = Programs; Prentice—Hall, Inc. 1976.) is csak nehezen hozzáférhető. A könyv, mely a zürichi Műszaki Egyetemen (az ETH-n) tartott előadások összefoglalása, a PASCAL-nyelvet használja a programok bemutatására. A könyv egyike a számítástechnika alapműveinek. Jó lenne gyorsan ismét kiadni. Ezzel különösen az egyetemi hallgatóságnak tennénk nagy szolgálatot.

egy „a[Ø]" komponenssel, hanem h darab újabb komponenssel fogjuk kiegészíteni. Ezért n darab tétel rendezéséhez az alábbi módon deklarált tömböt fogjuk használni:

var a: array [−h, . . n] ot tétel.

Ha tehát például egy n = 1ØØ tételből álló tömböt kívánunk rendezni úgy, hogy az első menetben az egymástól 49 „távolságra" lévő elemeket szeretnénk összehasonlítani (azaz, ha h[1]: = 49 értékkel dolgoznánk), akkor a rendezéshez 1ØØ + 1 + 49 = 15Ø tételnek megfelelő méretű tömbnek kellene a tárban helyet biztosítani.

A h [1 . . t] tömbben tárolt lépések optimális meghatározása néhány igen nehéz matematikai problémát vet fel. Nem ismeretes, hogy milyen sorozat adja a legjobb eredményt. Knuth a következő sorozatot javasolja:

. . . , 121, 40, 13, 4,1

Ajánlja még a

. . . , 31, 15, 7, 3, 1

sorozatot is.

Az alább bemutatott programban a

. . . , 9, 5, 3, 1

sorozattal fogunk dolgozni. Íme a program (helyesebben a procedúra — mint a PASCAL szintaxisából ismeretes, a procedúra csak egy program keretében fut le, önállóan nem futtatható —!):

```
procedure Shell-rendező;
    const t=4; {négy „menetben" fogunk
               rendezni}
    var i, j, k, s: index;
        x: tétel; {az előző példából ismert
           módon}
        m: 1 . . t; {a h tömb indexe}
        h: array [1 . . t] of integer;
        {h tartalmazza az egymás utáni rendező menetekben alkalmazott „távolságokat"}
begin {először a h tömb elemeinek adunk értéket}
    h [1] := 9; h [2] := 5; h [3] := 3; h [4] := 1;
    for m := 1 to t do {a különböző távolságok beállítása}
    begin k := h [m]; s := −k; {a strázsa helye}
        for i := k + 1 to n do
        begin x := a[i]; j := i − k;
            if s = Ø then s := −k; s := s + 1;
            a [s] := x;
            while x. kulcs < a [j]. kulcs do
                begin a [j + k] := a [j]; j := j − k
                end;
                a [j + k] := x
        end
    end
end
```

Akik a µMagazin múltkori számában olvasottakon felbuzdulva szerencsésen meg tudták szerezni D. Miller könyvét a C—64 adat-fájljairól, azok a „Home inventory System" nevű nagyobb programban megtalálhatták a fenti program egyik módosított változatának BASIC-ben írt megfelelőjét, mely így néz ki:

```
25ØØØ rem** — Shell—Metzner-rendező —
          szubrutin —**a rendezendő füzérek (tételek) az a$ elnevezésű változóban vannak. A tételek száma: n
25Ø1Ø m = n
25Ø2Ø m = int (m/2): rem felezem a sorozatot
25Ø3Ø if m = Ø then 251SØ: rem VÉGE
25Ø4Ø j = 1 : k = n − m
25Ø5Ø i = j
25Ø6Ø z = i + m
25Ø7Ø if a$ (i) < a$ (z) then 251 2Ø
25Ø8Ø t$ = a$ (i):a$(i) = a$ (z): a$ (z) = t$
25Ø9Ø i = i − m
251ØØ if i < 1 then 251 2Ø
2511Ø goto 25Ø6Ø
2512Ø j = j + 1
2513Ø if j > k then 25Ø2Ø
2514Ø goto 25Ø5Ø
2515Ø return
```

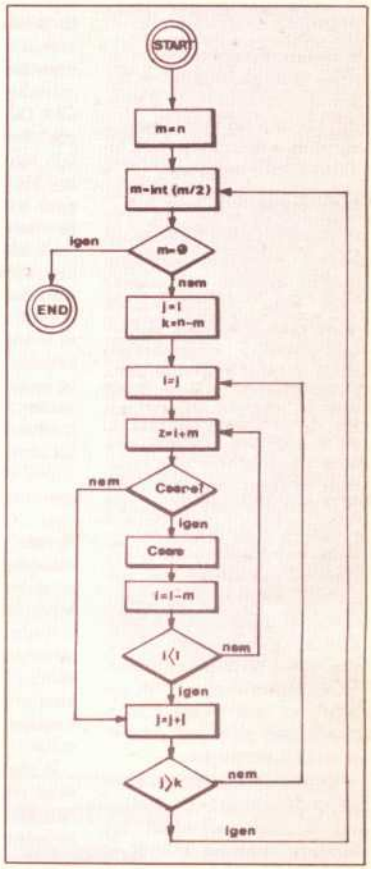
A működés magyarázata helyett álljon itt az algoritmus folyamatábrája. Ezzel együtt jó kis szellemi torna lesz a Shell—Metzner-rendező működésének megértése. Az algoritmus a „távolságot" így állítja be PASCAL-ra fordítva:

```
.
.
.
m := n; {ahol n a rendezendő tételek száma}
while m ≥ 1 do
    begin h [i] := m div 2;
          i := i + 1
end;
t := i;
.
.
.
```

A h tömb ilyen feltöltése N. Wirth szerint nem a legjobb. Ezt írja ugyanis könyvében: talán meglepő, hogy optimális esetben a növekmények nem lehetnek egymás többszörösei.

Ennek ellenére a bemutatott rendező algoritmust bátran ajánlhatjuk C—64 tulajdonosok részére.

Érdekes befejezésül egy pillantást vetni arra a táblázatra, melyben N. Wirth összefoglalja a különböző rendező algoritmusokkal végzett végrehajtásiidő-mérések eredményét:



A „Shell—Metzner-sort" algoritmus folyamatábra

Az algoritmus neve	A kiindulási állapot					
	rendezett		véletlen		fordított	
Közvetlen beszúrás	12	46	366	1129	704	2150
Bináris beszúrás	56	76	373	1105	662	2070
Shell-rendező	58	186	127	373	157	435
Gyors-rendező	31	55	60	137	37	75

A táblázatban megadott számok a futási időt mutatják n—256 tétel rendezésére egy CDC 66ØØ-as gépen ezred másodpercben. A bal oldali szám csak kulcsok rendezésére, a jobb oldali kulcsot és adatokat is tartalmazó tételekre vonatkozik.

Itt is látszik, hogy a bináris beszúrás alig javít a közvetlen beszúráson.

Az igen jónak bizonyuló „gyors rendezőt" az érdeklődők és a szerencsések megtalálhatják a könyvben.

— KE —

Sínstruktúrájú háló

Decentralizált vezérlésű síntopológiájú helyi hálózatok esetén az alábbi hálózatelérési módszerek terjedtek el

— CSMA/CD
— Round robin
— Vezérelt jelzéses (token bus)

A síntopológiájú hálózatra az egyes munkaállomások párhuzamosan csatlakoznak. A hálózati kábelt a két végén illesztve zárják le a zavaró reflexiók kiküszöbölése érdekében.

A hálózati kábel lehet koax vagy sodrott érpár. Az ilyen struktúrák közös jellemzője, hogy a hálózaton egyidőben csupán egyetlen adatcsomag lehet jelen. A sínhálózatok egyszerűen bővíthetők (átkonfigurálhatók), megbízhatóságukat a kábel folytonos volta nagymértékben növeli (nem kell a kábelbe állomásonként aktív elemeket közbeiktatni, mint a gyűrűs hálózatok esetében), az egyes állomások hibája nem teszi üzemképtelenné a rendszert. A sínstruktúrájú hálózatok leggyakrabban véletlenszerű hálózateléréssel működnek. Az ilyen jellegű hálózatelérési módszereket a hálózathoz való hozzáférés, az üzenetcsomag-ütközés felismerése, az üzenet megismétlése, valamint az üzeneteknek a hálózatból való eltávolítása szempontjából vizsgálják.

Sínstruktúrájú hálózatokban léteznek időosztásos rendszerek is, ezek működését egy központi ütemező időzíti, az egyes munkaállomások üzenetei a hálózatban adott hosszúságúra tördelve kerülnek továbbításra (csomag), a sikeres átvitelt a vevőoldal egy időkorláton belül nyugtázza, ennek hiánya (negatív nyugta) jelzi az üzenetütközést, ilyenkor a megsemmisült üzenet forgalmazását meg kell ismételni.

Időosztás nélküli rendszerekben a továbbított üzenetek hosszúsága tetszőleges, az ilyen rendszer teljesítménye valamivel kisebb az időosztásosnál.

CSMA/CD

A CSMA/CD rendszer a „figyelj, mielőtt beszélsz" elvet valósítja meg, azaz a forgalmazni kívánó állomás forgalmazás előtt „befigyel" a csatornába és csak akkor kezd el forgalmazni, ha a csatornát szabadnak találja, nehogy az üzenetek megsérüljenek. Innen származik az eljárás neve is, CD: Collision Detection = ütközésértékelés. Ha a vonal foglalt, úgy többfélét is tehet, pl. egy kis kivárás után újra befigyel a vonalba. A megoldás hátránya, hogy a kivárások miatt a csatorna átbocsátóképessége csökken, előnye, hogy az ütközések száma csökkenthető (ha két üzenet egyszerre kerül rá a közös átviteli közegre — kábelre — azok ott ütköznek és megsérülnek, ezért mindkét üzenet forgalmazását meg kell ismételni). A megoldás ott alkalmazható előnyösen, ahol az átvitel lassabb, mint amit a rendszer egyébként megenged.

Léteznek olyan rendszerek is, melyek a vonalfoglaltság észlelése utáni kivárás idejét valamilyen algoritmus (szabály) szerint változtatják (pl. egy-egy sikertelen kísérlet után növelik a kivárás idejét), illetve egy meghatározott számú kísérlet után abbahagyják a sikertelen forgalmazási kísérletezést.

A csatornafoglaltságot alapsávú rendszerekben a csatornán forgalmazáskor jelenlevő impulzussorozat, szélessávú rendszerekben a vivőfrekvencia megléte jelenti, az ütközés esetén a csatornán feszültségcsúcs jelenik meg, melynek nagyságát a kábelhossz is befolyásolja; ez a tény adott esetben a maximális kábelhosszat is korlátozhatja.

Szélessávú rendszerekben az adatcsomagok ütközését az elküldött adatcsomag, illetve a visszakapott nyugta bitenkénti összehasonlításával érzékelik.

Round robin eljárás

Az ilyen elven működő rendszerben az egyes munkaállomások adási, illetve vételi pontjaikon kapcsolódnak egymáshoz. Működésük során az állomások lehetnek:

— tétlen állapotban (nem akar forgalmazni)
— a forgalmazás megkezdésére váró állapotban
— a forgalmazás folytatására váró állapotban.

A forgalmazásra váró állomások csatornakérőjelet tesznek a csatornára, majd „befigyelés"-sel ellenőrzik annak foglaltságát. A csatorna szabaddá válása után a vonalat, ezzel a forgalmazás jogát mindig a legkisebb indexű állomás kapja meg (a magasabb indexű állomás érzékeli a kisebb indexű csatorna kérőjelét, így a csatornát foglaltnak érzékeli.)

Az ismertetett eljárásnak létezik egy prioritáson alapuló változata is, ebben az egyes állomások különböző prioritási szintekhez vannak hozzárendelve és a csatorna-hozzáférés jogát a forgalmazásra váró állomások közül mindig a legmagasabb prioritási szinthez tartozó kapja meg.

Vezérelt jelzés (token bus)

A sínre az egyes munkaállomások egy telepítés által meghatározott sorrendben (fizikai sorrend) kapcsolódnak rá, de ugyanakkor minden állomás rendelkezik egy logikai azonosítóval is (a kettő nem szükségszerűen azonos). Az egyes állomások hálózathoz való hozzáférését azok logikai sorrendje szerint meghatározott időzítésű hozzáférési jog szabályozza (token). A hálózat úgy működik, hogy a vezérelt jelzést birtokló állomás adott ideig birtokolja a hálózatot, ezalatt forgalmazhat, majd ezt követően a hálózat birtoklásának joga továbbkerül a következő logikai című állomásra.

A hálózati csatolóegység

A helyi hálózat munka- és server állomásai a hálózati közegre (kábel) egy csatolóegység segítségével csatlakoznak. A csatolóegység a munkaállomás (személyi számítógép vagy képernyős terminál) belső buszára vagy szabályos interfészére (V24) csatlakozik. A személyi számítógépes helyi hálózatokban az első megoldás az elterjedtebb, itt a csatolóegység nyomtatott áramköri kártya kivitelű és magába a PC-be dugaszolják be.

A csatolóegység nagyintegráltságú elemeket (speciális protokolvezérlő áramkörök, mikroprocesszorok, nagykapacitású tárelemek) tartalmaz, működését ROM-okba égetett mikroprogram vezérli.

A csatolóegység feladata a közvetlen hálózati forgalomvezérléssel kapcsolatos funkciók ellátása, pl. az adás, illetve vételirányú csatornakapcsolat megvalósítása, vonalfoglaltság-figyelés, az adásismétlés szervezése, nyugtázás, címfelismerés, az üzenetcsomag összeállítása, adatpufferelés, adatátvitel a PC, illetve a hálózat felé.

Ezen funkciók megvalósításával a csatoló tulajdonképpen a PC központi egységét tehermentesíti a perifériákus funkciók ellátása alól.

Az adatátvitel a helyi hálózatok esetén is az OSI nyílt architektúra 7 rétegű modelljének megfelelően valósul meg, a modell alsó 2—4 (fizikai, adatkapcsolati, hálózati, szállítási) rétegfunkcióját a csatolóegységben valósítják meg.

Érdekesebb alkalmazások

A világon jelenleg a legtöbb helyi hálózatot irodai és ügyviteli rendszerekben alkalmazzák, de jelentős számban találhatók helyi hálózatok ipari alkalmazásokban is. Mivel az irodai alkalmazások a közismertebbek, a cikk inkább né-

ÓZATOK

k elérési módszerei

hány érdekesebb, megvalósított ipari alkalmazást mutat be, ezzel is szemléltetve a helyi hálózatok sokoldalú alkalmazhatóságát.

A General Motors 15 részlegét kapcsolta össze helyi hálózat segítségével. A hálózattal futószalagot, szerelőrobotokat vezérelnek, továbbá ezen hálózattal oldották meg a szereléshez szükséges alkatrészek teljes nyilvántartását is.

A General Electric rendszere irodai, tervezést segítő és gyártási alhálózatokból áll. Az irodai rendszer közismert feladatokat lát el, a tervezést segítő rendszerrel gyártmánytervezést végeznek, ennek leterheltsége a rendszer használatából kifolyólag lökésszerű és benne a gépi válaszidők is ennek megfelelően alakulnak (50—500 ms). A gyártási rendszerrel robotokat, különböző gépeket és kamerákat kapcsolnak össze, a hálózatot a fenti funkciókkal egyidőben hangátvitelre is használják. A rendszer gépi válaszideje szélsőséges körülmények között 5—5000 ms között változhat.

Az egyes rendszerek esetében jelentős szerepe van a teljesítmény/ár hányadosnak, ami a legmegfelelőbb hálózatelérési módszer megválasztásával is befolyásolható, ez is az egyik oka a CSMA/CD eljárás közkedveltségének (egyszerű, hatékony műszaki megoldás, készen kapható CSMA vezérlő tokok stb.). Ezt a megoldást alkalmazza egyébként a General Electric, A Data General, a HP és a Siemens is rendszereiben.

Az ipari alkalmazásbeli hálózatok zömmel szélessávúak és sinstruktúrájúak, a hálózat az egész intézmény területére kiterjed, továbbá a fő funkciók (folyamatvezérlés) mellett számos egyéb szolgáltatást is nyújtanak az üzemeltetőknek (kép- és hangátvitel). Az ilyen rendszereknek további előnye, hogy a főhálózat a kisebb alhálózatokkal egyszerűen összekapcsolható komplex rendszerekké. Az ipari alhálózatok általában alapsávúak és 4-10 csomópontot (munkaállomás) tartalmaznak.

Az alhálózatokból felépített struktúra azért előnyös, mert vele a felhasználói igények egyszerűbben elégíthetők ki.

Az elkövetkezendő időszakban kialakítandó munkahelyeken — főleg valós idejű alkalmazásokban — várhatóan a nagysebességű adatcsatornák mellett több kisebb sebességű csatornát is tartalmazó hálózatokat fognak üzemeltetni. Ipari rendszerek esetében igen jelentős szempont a rendszer jól definiált válaszideje, főleg valós idejű alkalmazásokban, ilyen esetben célszerűbb a vezérelt jelzéses rendszer alkalmazása. Léteznek olyan rendszerek is, melyekben a forgalomtorlódás soha sem fordul elő, így állandóan maximális teljesítménnyel üzemelnek, a válaszidők minden esetben jól definiáltak. Az IBM ipari alkalmazásokra készített rendszere sinstruktúrájú, irodai alkalmazásokra készített hálózata gyűrűs struktúrájú. Az IBM megjelenése a helyi hálózatok területén egyúttal a megoldás „defacto szabvány"-nyá válását is jelentheti.

Ami az elkövetkezendő 4—5 év várható perspektiváját illeti, a kapcsolatteremtésben elsősorban a vállalatvezetés, az egyes részlegek és a vállalati nagyszámítóközpont összekapcsolása várható főleg alapsávú helyi hálózatokkal. Az alkalmazások terén elsősorban a megrendelés-felvétel, megrendelés-nyilvántartás, raktárkészlet-nyilvántartás, valamint a termelésütemezés „hálózatosítása" várható leginkább; főleg nagyobb intézményeknél ezekhez a rendszerekhez csatlakozhatnak a különböző géppel segített tervező-gyártó rendszerek (CAD—CAM), illetve az irodai apparátus munkáját kiszolgáló alrendszerek is.

CSEH KÁLMÁN

Tervezőintézet
országos
számítógépes
grafikai
nyilvántartási
rendszerek
fejlesztéséhez
keres
rendszerszervező,
tervező
és programozó
szakembereket.
Jelentkezés:
569-122/218-as mellék

Fejlesztési Intézetünkbe felvételre keresünk korszerű mikroprocesszoros terminálok és berendezések HARDWARE—SOFTWARE-fejlesztéséhez villamosmérnököket, készülék konstrukciós kialakításához felsőfokú végzettségű szakembereket.

RUGALMAS MUNKAIDŐ
KIEMELT FIZETÉS
MAGAS JÖVEDELEM

Jelentkezés:
TELEFONGYÁR
1143 Bp. Hungária krt. 126-132.
Tel.: 634-330.

DÖNTÉSÉT MEGKÖNNYÍTJÜK!

Széles választék, kedvező ár, sokoldalú szolgáltatás!

8 bites ALAPKONFIGURÁCIÓK

PROPOS—8 (CP/M vagy ezzel kompatíbilis) operációs rendszerrel

M08X	250 000,— Ft
PROPER—8	250 000,— Ft

A konfigurációk érdekeltségi alap nélkül megvásárolhatók

16 bites ALAPKONFIGURÁCIÓK

PROPOS—16 (IBM PC XT vagy ezzel kompatíbilis) operációs rendszerrel

PROPER—16/m	266 000,— Ft
PROPER—16/W1	439 000,— Ft
PROPER—16/W2	549 000,— Ft
PROPER—16/MEGA	699 000,— Ft
MÁTRIXNYOMTATÓ (MP—80, kvázigrafikus jelkészlettel)	30 000,— Ft
PROPOS—16 OPERÁCIÓS RENDSZER	29 000,— Ft

RENDELÉSEKET VESZÜNK FEL 1986. II. FÉLÉVRE

PROPER—16/MT (IBM PC AT kompatíbilis operációs rendszer)
850 000,— Ft

PROPER—16/G (nagy teljesítményű grafikus munkahely)
1 500 000,— Ft

SZOLGÁLTATÁSOK:

12 hónap garancia, további 12 hónapra a hardver árának 6%-áért hardver-szervizszolgálat
Világszínvonalú perifériák
Széles körű szoftverválaszték
Kulcsrakész alkalmazói rendszerek
Hálózatba integrálási lehetőség
Országos szervizhálózat

KÉRJE RÉSZLETES ÁRJEGYZÉKÜNKET!

Szki

Számítástechnikai Kutató Intézet és Innovációs Központ
Budapest 1251 Pf. 19.

Információ:
SCI—L Számítástechnikai Informatikai Fejlesztő Leányvállalat
Budapest, Iskola u. 10. 1011
Telefon: 153-204

AZ SZKI STABIL PARTNER!

SCITEL Számítástechnikai Fejlesztő Leasing Leányvállalat
Budapest, Donáti u. 35—45. 1015
Telefon: 350-180/142

PTA–4000

ROM–leírás

A gép alapkiépítésben az MØ lapon az AØØØ—FFFF címtartományban tartalmaz ROM-ot. Ezt cím szerinti sorrendben tárgyaljuk.

Az AØØØ—AØ1F címterületen a kiíratáshoz használt 1. segédtábla található. Ezt a táblát a tábla címmutatója követi az AØ2Ø—AØ7E címterületen. A 2., ugyanilyen célú segédtábla a folytatás az AØ7F—A28A címterületen.

A következő nagy terület a gép alapműködését kiszolgáló szubrutinoké az A28B—AFFF címterületen.

A BØØØ—BØE9 az 1. kulcsszótáblázat helye. A táblázat belsejében jelentős nagyságú „üres" hely található, valószínűleg a későbbi bővítések céljára fenntartva.

Ezt egy rövidebb, újabb rendszerprogramokat tartalmazó rész követi a BØEA—B152 címtartományban.

BASIC szubrutinok I.

A kulcsszavakhoz tartozó 1. szubrutincsoport a B153—B7FF címterületen található. Az itt levő szubrutinok kezdő- és végcímeit az *1. táblázatban* ismertetem.

A folytatás a 2. kulcsszótábla a B8ØØ—B887 címterületen, ismét jelentős „üres" résszel.

Ezt az ún. ugrótábla követi. Bizonyos szubrutinok ezeken keresztül, kétlépcsős ugrással érhetők csak el. A címterület: B888—B8A5.

BASIC szubrutinok II.

A B8A6—BBD5 területen újabb kulcsszavakhoz tartozó szubrutinok vannak, melyek kezdő- és végcíme a *2. táblázatban* található.

Ezután néhány, a kazettával dolgozást szolgáló szubrutin következik. A BBD6—BBF4 címterületen a szalagon az ún. „fejrész"-t előállító rész, a BBF5—BCE7 között a parancsok kivitelét/beolvasását vezérlő szubrutin, a BCE8—BD3B területen a „fej" kiíratót/beolvasót, a BD3C—BDA3 területen az állományt kiíratót szubrutinok vannak.

A BDA4—BDCB területen egy szubrutindarab található, bizonyára csak így fért el. A BDCC—BDE6 területen az egy bájtot kiíratót szubrutin, majd a BDE7—BDEF területen ismét egy szubrutindarab, a BDFØ—BDF8 területen az egy bájtot beolvasó szubrutin lelhető fel.

Ezután a külső készülékeket vezérlő parancsok következnek. Először az RMT a BEF9—BF1Ø címterületen, ezt követik a külső készüléket be-(REMOTE ON), BF11—BF42 és kikapcsoló (REMOTE OFF), BF43—BF75 parancsok, majd újra egy szubrutindarab: BF76—BFFF.

3. kulcsszótábla

Néhány rövidebb szubrutin következik; az Xreg töltése a 78F1 címen tárolt értéktől függően CØØ1—CØØD (a CØØØ címen egy állandó található), az Ureg töltése a 7883/7884 és a (7AØ7) 7885 címeken levő értéktől függően CØØE—CØ1C, a TRACE-vektor, CØ1D—CØ1F. Ezt a parancsok kezdőcímtáblája követi CØ2Ø—CØ53, szintén jelentős bővítési lehetőséggel.

A 3. kulcsszótábla az ezután következő rész a CØ54—C34E címterülettel.

Ismét rövidebb csoportokat találunk ezután. A BASIC üzenetei C34F—C36A, a DEF-billentyűk táblázata C36B—C385, egy adat C386—C38A, a vezérlőkódok (ØØ—ØF ASCII-érték) ugrótáblája, C38B—C3AA, egy általam nem ismert célú táblázat: C3AB—C3FF.

Utasítások

Indítás:
C4ØØ.
A BASIC-fordító kulcsszavak az Ureg-ben:
C4Ø1—C457.
Ezt követik az utasítások szubrutinjai, melyek kezdő- és végcímei a *3. táblázatban* találhatók:
C458—C9E3.
Az EDITOR-hidegindítás követi ezt:
C9E4—CA54.
A CLEAR ALL (CA) billentyű szubrutin belépőpontja:
CA55—CA57.
EDITOR-melegindítás (RUN-módban):
CA58—CA63.
CL-billentyű szubrutin:
CA64—CA79.
EDITOR-ugrás a mutató törlése nélkül:
CA7A—CA7F.
EDITOR-ugrás:
CA8Ø—CB6Ø.
RCL-billentyű szubrutin:
CB61—CB68.
A RESERVE-mód bekapcsolása:
CB69—CB9B.
A SHIFT-billentyű bekapcsolása:
CB9C—CB9F.
A MODUS-billentyű bekapcsolása:
CBAØ—CBC6.
Az INS-billentyű bekapcsolása:
CBC7—CBCE.
A DEL-billentyű bekapcsolása:
CBCF—CBE3.
A CURSOR RIGHT-billentyű bekapcsolása:
CBE4—CC21.
A CURSOR LEFT-billentyű bekapcsolása:
CC22—CC37.
A CURSOR UP-billentyű bekapcsolása:
CC38—CC47.
A CURSOR DOWN-billentyű bekapcsolása:
CC48—CCCØ.
Az ENTER-billentyű bekapcsolása:
CCC1—CD7Ø.
Az OFF-billentyű bekapcsolása:
CD71—CD88.
Az ERROR 1-üzenet ugráshelye:
CD89—CD8A.
Az ERROR-üzenetek (a kódok az Ureg-ben):
CD8B—CDE5.
Egy karakter beadása az INPUT PUFFER-be INSERT-módban:
CDE6—CEØF.
Egy karakter beadása az INPUT PUFFER-be normál módban:
CE1Ø—CE37.
Egy karakter törlése az INPUT PUFFER-ből:
CE38—CE9E.
A rendszer által lefoglalt terület kezdőcímének meghatározására szolgáló szubrutin:
CE9F—CFCA.
A rendszercímek inicializálása:
CFCB—D02A.
Az INPUT PUFFER törlése:
D02B—D03D.
A rendszerüzenetek mutatóinak meghatározására szolgáló szubrutin:
D03E—D064.
BASIC veremműveletek szubrutinja:
D065—D07Ø.
Az Ureg tartalmának betöltése a BASIC verembe:
D071—D07F.
Változók törlése:
D08Ø—D0D1.
Az ún. AR—X és AR—Y tartalma közt fennálló egyenlőség vizsgálata:
D0D2—D0F8.
Két karaktersorozat egyenlőségvizsgálata:
D0F9—D2E5.
Egy adott sorszámú sor kikeresése:
D2E6—D3D4.
Decimális/hexadecimális konverzió:
D3D5—D406.
Kódolt változónevek meghatározása:
D407—D45C.
Az Yreg-t mutatónak használva a változót kereső szubrutin:
D45D—D46Ø.
Az Ureg-ben tárolt változók kezdőcímének meghatározása:
D461—D5F8.
Egy 16 bites paraméter Uregbe töltése:
D5F9—D6AC.
Egy adat:
D6AD—D6BF.
A változó átvitele az AR—X-ből vagy az INPUT PUFFER-ből:
D6CØ—D6DE.
Az AR—X feltöltése egy változóval:
D6DF—D924.
Két karaktersorozat összekapcsolása:
D925—D992.
PEEK/PEEK#:
D993—D9A9.
INKEY $:
D9AA—D9BØ.
CHR$:

D9B1—D9CE.
STR$:
D9CF—D9D6.
VAL:
D9D7—D9DA.
A ROM további részeinek ismertetését folytatjuk.

DR. SIMONYI ENDRE

1. táblázat

Megnevezés	Címterület
SORGN	B153—B159
ROTATE	B15A—B169
COLOR	B16A—B17F
CSIZE	B180—B190
GLCURSOR	B191—B1B3
LF	B1B4—B200
LCURSOR/ TAB	B201—B221
LINE	B222—B223
RLINE	B224—B2EB
LPRINT	B2EC—B753
LLIST	B754—B7FF

2. táblázat

Megnevezés	Címterület
CSAVE	B8A6—B8F8
CLOAD	B8F9—B993
MERGE	B994—BB55
CHAIN	BB6A—BBB8
UP	BBC0—BBD5

3. táblázat

Megnevezés	Címterület
LET	C458—C4B5
STOP	C4B6—C50C
INPUT	C8FA—C967
LOCK	C968—C969
UNLOCK	C96A—C96D
LIST	C96E—C987
DIM	C988—C9E3
END	C50D—C514
GOTO	C515—C5B3
IF	C5B4—C5DF
ON	C5E0—C64D
GOSUB	C64E—C675
REM	C676—C67B
USING	C67C—C683
DATA/ ARUN/ AREAD	C684—C68B
TRON	C68C—C692
TROFF	C693—C696
DEGREE	C697—C6A3
RADIAN	C6A4—C6A7
GRAD	C6A8—C6AB
RETURN	C6AC—C703
NEXT	C704—C710
FOR	C711—C776
POKE	C777
POKE #	C778—C7A1
RESTORE	C7A2—C7B7
READ	C7B8—C809
NEW	C80A—C85E
CLEAR	C85F—C862
CALL	C863—C8B3
RUN	C8B4—C8C6
CONT	C8C7—C8F9

16 k-ról 17 k-ra

A ZX81 bővítése

A ZX81 alapgépnek 1 k vagy 2 k belső memóriája van. A belső memória a 16 k-s bővítő felhelyezésekor lekapcsolódik, mert a RAM C.S vonal H szintre (+5 V) kerül. Egyszerű átalakítással elérhető, hogy ez a belső memória használható legyen a 16 k-s bővítő rákapcsolása után is úgy, hogy a gép akkor is működőképes marad, ha a bővítő nincs rákapcsolva.

Az átalakítás egy címdekóder és néhány kapu (két IC) beépítéséből áll. A címdekóder (SN 74LS139) azért szükséges, hogy ha a 16 k-s bővítőt rákapcsoljuk, akkor a belső memóriaterület más címet kapjon. A kapuk (SN 74LS00) feladata, hogy a belső RAM az „eredeti" formában akkor is működni tudjon, ha a bővítés nincs rákapcsolva.

A ZX81 belső címdekódolása miatt a 8 k-nyi ROM 16 k címtartományt foglal el, mert olyan a dekódoló rendszere, hogy a 8 k kétszer jelenik meg. A második 8 k címtartományt (árnyék ROM) a BASIC nem használja. Így ha ide címezzük a belső memóriát, az a RESET (RESET vonal L-re kapcsolása) hatására nem törlődik.

A bemutatott áramkör megoldja a ROM és a belső RAM címdekódolását. A *rajzon* megtalálható a gép hátulján levő csatlakozósáv (EDGE CONNECTOR) bekötése is. Segítségével elég könnyen megkereshetők az átalakításhoz szükséges vonalak. Az x-szel jelölt helyeken a fóliát át kell vágni.

Az áramkör működése

— ROM C.S. előállítása az első 8 k területen (0—1FFF h),
— RAM C.S. előállítása a belső memória számára.

A belső RAM bővítés nélkül két címtartományban jelentkezik: egyszer az eredeti helyen és egyszer az új helyen a 2000 h—2FFF h tartományban abban az esetben, ha a 3. kapu bemenete $\overline{Q}_2$-re van kötve. Ha $\overline{Q}_3$-ra van kötve — szaggatott vonal —, akkor a 3000 h—3FFF h tartományban jelentkezik. Ez mind a két esetben 4 k tartomány, így a nem teljes címdekódolás miatt a belső memória 1 k esetén négyszer és 2 k esetén, tehát ha 2 k van a gépben, kétszer fordul elő egymás után a címtartományban.

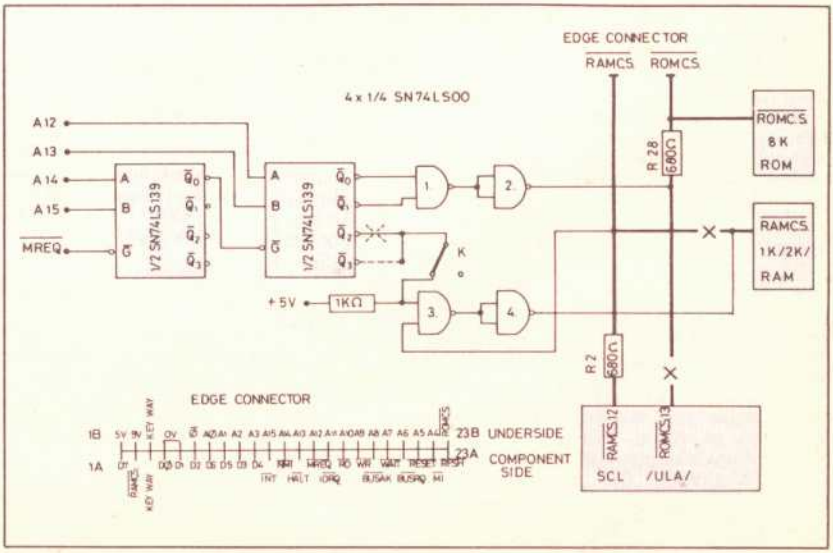
Bővítés esetén, mivel a bővítő a csatlakozósáv RAM C.S. vonalát H-ra kapcsolja, a belső RAM már csak az új helyen jelenik meg, az eredeti helyén nem, így nem zavarja a bővítő működését.

Az átépítéshez lehetőleg gyakorlott amatőrök fogjanak hozzá, mert egy hibás forrasztás vagy fóliaátvágás tönkreteheti a gépet.

Nagyobb, 32 k, 64 k memóriabővítők használhatják azt a címtartományt, ahová a belső memóriát címeztük, ezért ezekkel a memóriabővítőkkel az átalakított gép csak úgy használható, ha a k kapcsolót megszakítjuk!

Az áramkör más jelű, de hasonló funkciókkal rendelkező IC-kből is felépíthető. A helyes lábbekötést a típusnak megfelelően katalógusból kell kikeresni.

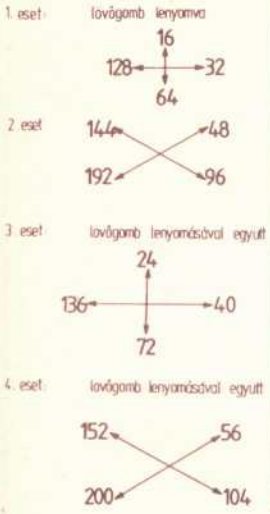
KOVÁCS SZILVESZTER



PRIMO

Botkormány

A botkormánykezelő rutin *(1. lista)* a 16443-as tükörregiszteren keresztül lépteti a botkormány számlálló áramkörét (lásd *ábránkat*). Ugyanezen a tükörregiszteren keresztül olvassa be a dekódoló áramkör helyzetét, és az eredményt az 50000-es decimális címre helyezi. Ha két botkormányt használunk, akkor a második lekérdezésének eredménye az 50001-es címen található.



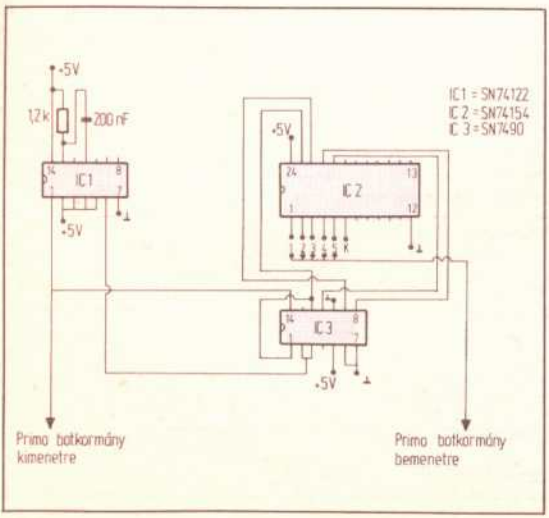
A rutin alkalmas arra is, hogy két nyomógomb egyidejű megnyomása esetén további irányításra alkalmas értékeket kapjunk.

A Rajz botkormánnyal nevű programban *(2. lista)* a DATA-k tartalmazzák a botkormánykezelő rutint. A 200-as sorban történik a rutin hívása. A lekérdezés eredményét a 210-es sorban kapjuk meg. Ettől függően vezérelhetjük négy irányba a rajzoló képpontot. A program a lövőgombot nem használja, de annak megnyomása esetén k=8 értéket kapunk. Ezzel további vezérlési lehetőségekhez jutunk.

GERENDÁS SÁNDOR

```
 1                              ORG   40000T
 2                              LOAD  40000T
 3 9C40 FD213B40                LD    IY.16443T
 4 9C44 0E08                    LD    C.8
 5 9C46 2150C3                  LD    HL.50000T
 6 9C49 3600                    LD    (HL).0
 7 9C4B FD7E00                  LD    A.(IY+0)
 8 9C4E CBBF                    RES   7.A
 9 9C50 CBF7                    SET   6.A
10 9C52 D300                    OUT   (0).A
11 9C54 FD7700                  LD    (IY+0).A
12 9C57 FD4600          LEP:    LD    B.(IY+0)
13 9C5A CBB0                    RES   6.B
14 9C5C ED41                    OUT   (C).B
15 9C5E FD7000                  LD    (IY+0).B
16 9C61 DB64                    IN    A.(100T)
17 9C63 CB47                    BIT   0.A
18 9C65 37                      SCF
19 9C66 2801                    JR    Z.EGY
20 9C68 3F                      CCF
21 9C69 CB16            EGY:    RL    (HL)
22 9C6B 23                      INC   HL
23 9C6C CB57                    BIT   2.A
24 9C6E 37                      SCF
25 9C6F 2801                    JR    Z.KET
26 9C71 3F                      CCF
27 9C72 CB16            KET:    RL    (HL)
28 9C74 2B                      DEC   HL
29 9C75 FD4600                  LD    B.(IY+0)
30 9C78 CBF0                    SET   6.B
31 9C7A ED41                    OUT   (C).B
32 9C7C FD7000                  LD    (IY+0).B
33 9C7F 0D                      DEC   C
34 9C80 20D5                    JR    NZ.LEP
35 9C82 FD4600                  LD    B.(IY+0)
36 9C85 CBF8                    SET   7.B
37 9C87 FD7000                  LD    (IY+0).B
38 9C8A ED41                    OUT   (C).B
39 9C8C C9                      RET
40                              END
```

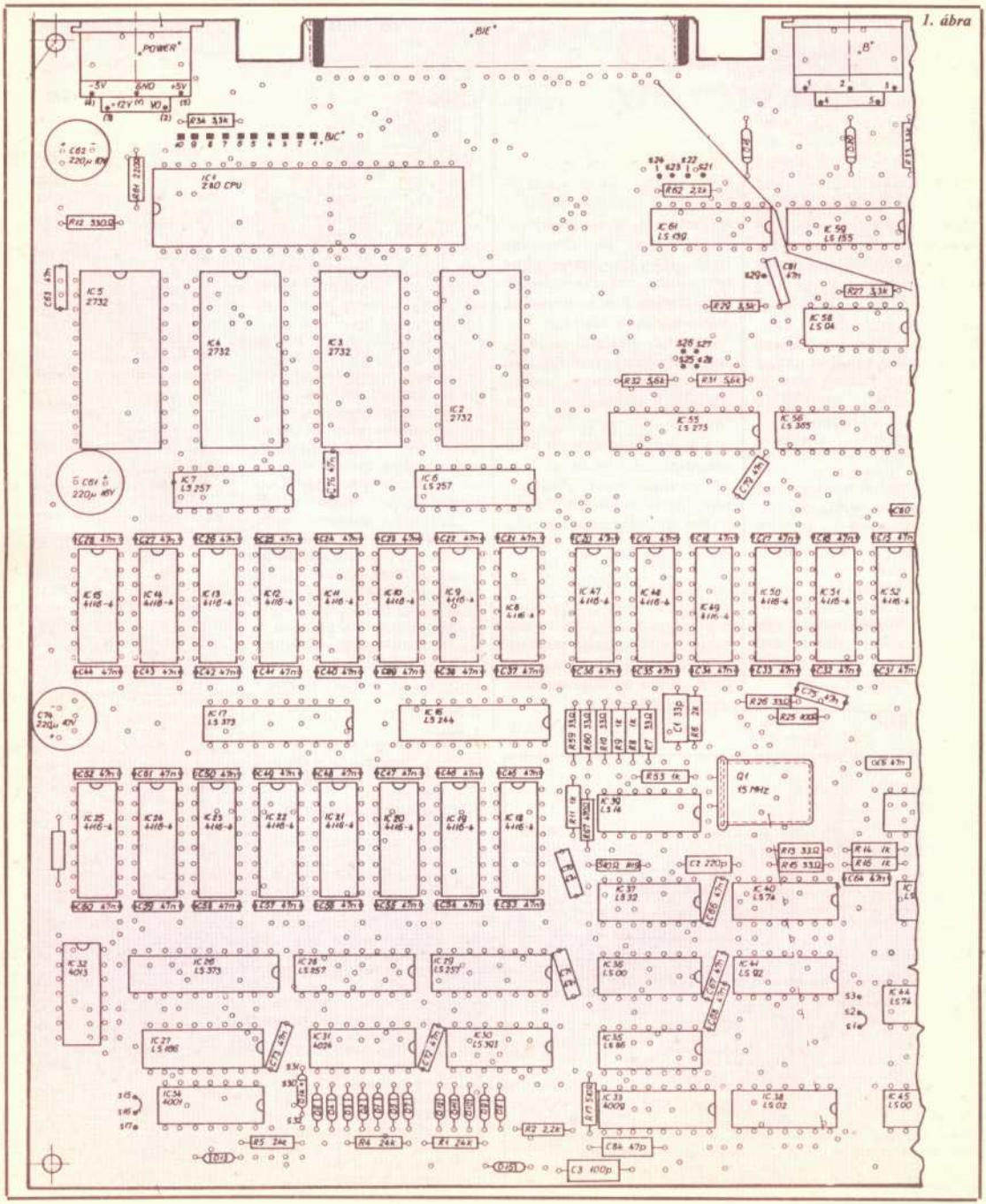
1. lista



2. lista

```
 10 REM Rajz botkormannyal
 20 FOR I=0T076
 30 READR
 40 POKE30000+I.R
 50 NEXT I
100 DATA253.33.59.64.14.8.33.80.195.54.0
104 DATA253.126.0.203.191.203.247.211.0
110 DATA253.119.0.253.70.0.203.176.237
115 DATA65.253.112.0.219.100.203.71.55
117 DATA40.1.63.203.22.35.203
120 DATA87.55.40.1.63.203.22.43.253.70.0
125 DATA203.240.237.65.253.112.0.13.32
130 DATA213.253.70.0.203.248.253.112.0
135 DATA237.65.201
150 CLS
200 C=CALL(30000)
210 K=PEEK(50000-65536)
300 IFK=16.A=A+1
310 IF K=64.A=A-1
320 IF K=128.B=B-1
330 IF K=32.B=B+1
340 IFA>191.A=191
342 IFA<0.A=0
344 IF B>255.B=255
346 IFB<0.B=0
350 SET(B.A)
400 GOTO200
```

PRIMO


POWER
BIE
B
BIC
IC1 Z80 CPU
IC5 2732
IC4 2732
IC3 2732
IC2 2732
IC7 LS 257
IC6 LS 257
IC61 LS 139
IC59 LS 155
IC58 LS 04
IC55 LS 273
IC56 LS 365
IC17 LS 373
IC16 LS 244
IC25 4116-4
IC26 LS 373
IC28 LS 257
IC29 LS 257
IC32 4013
IC27 LS 186
IC31 4024
IC30 LS 393
IC34 4001
IC33 4009
IC38 LS 02
IC45 LS 00
IC39 LS 14
IC37 LS 32
IC40 LS 74
IC41 LS 92
IC44 LS 74
IC36 LS 00
IC35 LS 86
Q1 15 MHz
C2 220p
C3 100p
C84 47p
R2 2,2k
R5 24k
R4 24k
R1 24k
R62 2,2k
R27 3,3k
R29 3,3k
R32 5,6k
R31 5,6k
R34 3,3k
R12 330Ω
R26 33Ω
R25 100Ω
R53 1k
R13 33Ω
R15 33Ω
R14 1k
R16 1k
C61 220μ 16V
C74 220μ 40V
C62 220μ 10V

1. ábra

Príma ötlet

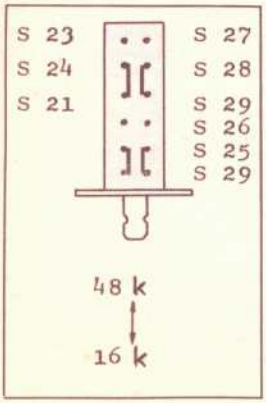
Minden számítógép-felhasználó életében elkövetkezik az a nap, amikor kinövi készülékét. Ekkor vagy komolyabb gépet vásárol, vagy a meglevőt bővíti.

Mi ez utóbbi mellett döntöttünk, amikor egy Primo A—32-es (16 k ROM, 16 k RAM) számítógépet kívántunk kibővíteni A—64-esre (16 k ROM, 48 k RAM). Először a javítással és átalakítással megbízott, Fehérhajó utcai Gelka szerviznél érdeklődtünk, ahol megtudtuk, hogy a bővítést hatezer (!) forintért végeznék el (ebből kb. 2000 forint az alkatrészár), ha lenne hozzá alkatrészük.

Maradt tehát a régi mondás: „Magad uram, ha szolgád nincs!" Íme a recept. Végy 16 db MCM 1618 vagy 4116 típusú integrált áramkört, 32 db 47nF-os tárcsakondenzátort és 1 db kétállású, 4 áramkörös izosztát kapcsolót. Az utóbbira azért van szükség, hogy a számítógépet a kapcsoló állásától függően 16 k vagy 48 k memóriával lehessen használni. Ha csak 48 k memóriával kívánjuk használni, felesleges a kapcsoló.

Az *1. ábrán* levő beültetésirajz-részleten feltüntetett (IC 8—15, IC 18—25) helyekre beforrasztjuk a 16 db IC-t és alájuk 1-1 db 47nF-os kondenzátort (C 21—28, C 37—44, C 45—52, C 53—60). Az A—32-es típus alapállapotában az S 21—S 24, S 25—S 29, S 28—S 29 pontok vannak összekötve. Bővítéskor ezeket oldjuk, és helyettük az S 23—S 24, S 25—S 26, S 27—S 28 pontokat zárjuk rövidre.

2. ábra



Az izosztát kapcsoló egyik lehetséges bekötését a *2. ábra* mutatja. A kapcsolót olyan helyre kell helyezni, hogy véletlen átkapcsolás ne következhessen be. *A bekapcsolt számítógépet ugyanis tilos átkapcsolni!!* A véleményünk szerinti legvédettebb helyet a számítógép jobb oldalán a billentyűzet íve alatt találtuk meg *(3. ábra)*.

Ha eddig eljutottunk, bedobozolhatjuk és bekapcsolhatjuk a gépet. A teljes képernyőt beteritő ponthalmaz jelzi, hogy jó munkát végeztünk.

Az A—32-es számítógép képernyője ugyanis a 26 624—32 767 címeken található. Memóriabővítéskor ez feljebb csúszik, pontosan 32 k-val. Az új képernyőhely meghatározását a gyártómű EPROM IC cseréjével végzi el. Mivel nekünk nem volt ilyen, más címeket tartalmazó IC-nk, szoftver úton kell megtanítanunk a számítógépet az új képernyő helyére.

Ezt kétféle módon tudjuk megtenni. Az egyik megoldás, hogy „vakon" a billentyűzet hangvisszajelzését figyelve begépeljük a következő közvetlen utasítássort:

POKE16458,232:
POKE16562,231:
CLEAR5Ø:CLS

és nyomunk egy RETURN-t. Hatására a képernyő megtisztul, és a bal felső sarokban megjelenik az „OK" felirat és a kurzor. Az egyik POKE a képernyő elejét, a másik a változómező elejét írja át.

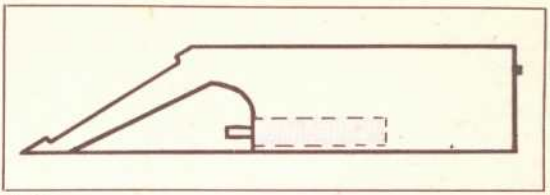
A másik megoldás, hogy a még 16 k-s gépbe beírjuk az alábbi programot, és *futtatás nélkül* magnóra kimentjük:

```
1Ø REM XXXXXXXXXXXX
2Ø REM X           X
3Ø REM X   48 K    X
4Ø REM X           X
5Ø REM XXXXXXXXXXXX

6Ø POKE16458,232:
   POKE16562,231:
   CLEAR5Ø:CLS

7Ø PRINT §Ø,Ø,CHR§(2)"PRI
   MO"CHR§(18)" BASIC
   SYSTEM '85.11."TAB(38)
   "48K"

8Ø DELETE1Ø—8Ø
```



3. ábra

Ebben az esetben az átkapcsolt gépbe már csak az I LOAD és a RUN parancsot kell „vakon" beírni, és elindítani a magnót. A sorszámmal ellátott LOAD parancs futtatásakor a számítógép automatikusan indítja a betöltött programot.

A RESET kapcsoló megnyomásakor a kurzor visszaugrik a régi helyére. Ekkor újra be kell írni a POKE16458,232-t, de már csak ezt, hogy a kurzor visszatérjen.

Tudjuk, hogy ez a tárbővítés elég kezdetleges, de több mint fél éve hibátlanul működik, és megnyugtat bennünket az a tudat, hogy pár ezer forint többletkiadást meg tudtunk takarítani. Amennyiben a leírtak megvalósításával bármilyen további kérdés, probléma merülne fel, szívesen állunk bárki rendelkezésére.

DEME ATTILA
HADNAGY ÁKOS

Ki ad magyarázatot?

A Primo számomra érthetetlen rendellenességeket produkál az adatok magnetofonszalagról történő beolvasása után.

1. Bizonyos programoknál a beolvasást követően READ—DATA utasításpárt használva, szöveges változóknál olyan jelenség lép fel, mintha minden READ utasítás után RESTORE utasítás lenne. Mivel a programban a hibát nem találom meg, ezekben az esetekben a következő „kerülő" megoldásokat használom:

a) READ utasítást csak a magnetofon használata előtt alkalmazok.
b) Az állandókat LET utasításban adom meg.
c) Minden adatot szalagról olvasok be.

2. Az adatok beolvasása közben FD ERROR jelzést kapok esetenként olyan programoknál is, amelyek már több alkalommal hibátlanul működtek.

Ebben az esetben csak egy kerülő megoldást találtam. Az adatok szalagra történő rögzítésénél a folyamatot az adatok képernyőre iratásával lassítom, és a CLOSE utasítás elé lassító ciklust írok.

SOMOGYI GYÖRGY

Azt mondják, hogy nyáron mindenki leáll, az emberek inkább a nyaralással törődnek, mint a munkával. Olvasóink levelei ezt az elméletet erősen cáfolják; azt hiszem ez a szokás az informatikával foglalkozók körében nem érvényes.

Fáy András, Miskolc,

Széchenyi u. 6. 3530

Azt szeretném megkérdezni, hogy Miskolcon miért nincs valamilyen számítástechnikai szakbolt, vagy valami ehhez hasonló. Miért minden Budapesten van. Az ország második városában rendkívül gyenge a számítástechnikai ellátottság. Szakkönyvekből tűrhető az ellátás, most nyílt egy bolt a Tanácsház téren a Novotrade gondozásában. Számítógép-ellátás borzasztóan gyenge, és az árak magasak. Az OFOTÉRT-ban egy C16 kb. 21 000 Ft, C64 a Bizományiban 30 000 Ft, VC—20 kb. 20 000 Ft, C116 24 000 Ft. Kazettás magnó COMMODORE-hoz 6000—10 000 Ft. Igen gyenge „fejlődést" lehet tapasztalni, annyit mindössze, hogy a CENTRUM-ban lehet kapni TVC-t 12 800 Ft-ért és ATARI800-at 20 000 Ft-ért. ZX81 kapható 5500 Ft-ért, ez sem túl olcsó.

A CENTRUM-ról olyan hírek terjedtek el, hogy a lakosság részére fog árusítani számítógépet (lásd pl. *µM* 1986/5. számát, 11. oldal).

Az egyik decemberi HÉT-ben láttam, azt hiszem, hogy az egyik illetékes azt mondta, szervezik C16 behozatalát kb. 50 dolláros áron.

Az újsággal alapjában véve meg vagyok elégedve, a hirdetés kicsit sok.

Levelét abban a reményben közlöm, hogy a miskolci számítógép-kereskedelemért felelős illetékesek is olvasni fogják.

Valkó Csaba, Miskolc,

Vörösmarty u. 66. VII./3. 3530

Nagyon örülök, hogy a lap az idei évtől havi formában jelenik meg. Ahogy olvastam az Olvasó írja rovatot, láttam, hogy az évi hat szám megjelenésével kapcsolatban nem csak nekem volt problémám. De ezek a problémák, mint ahogy Önök előre látták, egy csapásra megszűntek a havi megjelenéssel. De nemcsak ezért tartozom Önöknek köszönettel, hanem azért is, mert így minden hónapban hozzájutok kedvenc újságomhoz.

Látva a Magazinban a sok vitát a Magazin tartalmi és formai oldaláról tekintve, nagyon szoritok Önöknek, hogy olyan lapot tudjanak készíteni, amely valamennyi olvasó igényét ki tudja elégíteni. Nekem például nagyon tetszik a lap mostani formája, nem tudom, hogy egyeseknek miért vannak maximalista véleményeik a lappal kapcsolatban. Hiszen a lap úgy jó, hogy minél több dolog megtalálható legyen benne, ez pedig ilyen terjedelem (45-50 oldal) mellett lehetetlen feladat.

Mindenesetre várom a fejleményeket.

Mi is, az elismerést valamennyiünk nevében köszönöm.

Liptay László, Szabadkígyós,

Nefelejcs u. 13. 5712

Van egy Commodore VC—20-as számítógépem. Sajnos a memóriája csak 3,5 kbájt, és bővítőm nincs hozzá. Ezért a segítségét szeretném kérni. Ha lehetséges legyen szíves és küldjön nekem a géphez memóriabővítési rajzot, ugyanis szeretnék építeni egyet. A rajzért küldök egy kazettát.

A levelet átadtam az illetékes rovatszerkesztőknek. Kérését azért közlöm, hogy másoktól is kaphasson ötletet vagy segítséget.

Tóth István Gábor, Békéscsaba,

Tanácsköztársaság u. 52. I./5.

Sajnálattal meg kell állapítanom, hogy a korábban írt észrevételeket, melyeket a *µ*Magazin '85/6. számában megjelent C64 UNIN rutin programmal kapcsolatban írtam, az 1986/4. szám HELYREIGAZÍTÁS-a után ki kell egészíteni az alábbiakkal:

1. A közölt formában a stringekben szereplő „betűköz" karakterek nem egyértelműen értelmezhetők („„vagy""?). Ugyancsak hibás működésre vezet ha a melléklet szerinti 265-ös sorban a T$ cursor vezérlés helyett üres karakterre végez vizsgálatot.
2. Bizonyára nyomdahiba és nem szándékos elírás egy új változó a 140-es sorban a BI az egyébként szükséges B1 helyett.

Ugyanebben a sorban az AND B> helyett szedett AND D> már az igazán kezdők számára is annyira szembetűnő, hogy biztosan nem fogják szolgai módon „bepötyögtetni".

Mindezt figyelembe véve az összesen nyolc sor „HELYREIGAZÍTÁS" nyolc újabb hibát tartalmaz.

Úgy látszik ez a „szuper program" a nyomda ördöge miatt csak nem akar hibátlan lenni.

Egyébként biztosan igen jól hasznosítható program, különösen ha az alapfeladat néhány száz sor, és 30-40 különböző hosszúságú és fajtájú adatot kell a feldolgozás során bevinni.

A helyreigazítás helyreigazítása a következő:

```
105 T1=PEEK(51):T2=PEEK(52)
    :T6=T1+256*T2-41:A$=""+
    A$:T4=INT(T6/256):T3=T6-T4*
    256
140 B=LEN(A$):IFB1>15ANDB<6
    ANDB
    >ØTHENFORB=B+1TO6:
    J$=""+A$:GOSUB540:NEXT:GO
    TO16Ø
21Ø Z=(55296+L*4Ø+C):FORI=Z
    TOZ+P
    :POKEI,CO:NEXT:Y=1Ø24+
    L*4Ø+C:X=Y+P:IFB7<>7ORB
    =ØTHEN23Ø
265 POKES1,S3:S3=Ø:S4=8Ø:
    IFB8=7ANDT$="□"AN
    DA=1THENGOSUB535:
    J$=CHR$(14):GOSUB54Ø:A=Ø:GO
    TO195
34Ø A=A+1:GOSUB535:
    J$=LEFT$(A$,A-
    2)+T$+RIGHT$(A$,
    (AV-A+ABS(AV-A))/2):
    GOSUB54Ø:IFA>AVTHENAV=A
495 IFBØ<>7AND((B1=ØAND
    LEFT$(A$,1)="")
    OR(B1<>ØANDRIGHT$
    (A$,1)="""))THENA=1:AV
    =B+1:GOTO24Ø
5ØØ IFB6=ØTHENFORI=XTOYSTEP-
    1:
    POKEI,ASC(MID$(A$,1-X+B,1))
    AND63OR128:NEXT:GOTO195
54Ø L$=L$+"":POKE82Ø,PEEK(71):
    POKE821,PEEK(72):
    POKE(PEEK(82Ø)+PEEK
    (821)*256),41-LEN(J$)
```

A grafikus jelek jelentései az alábbi sorokban:

265 kurzormozgatás fölfelé
355 kurzor jobbra lép
36Ø kurzor jobbra lép
395 INSERT
415 kurzor balra lép
42Ø kurzor balra lép
425 HOME
43Ø CLEAR

Sajnos erre már semmi mentségünk nincs.

Szilvay Gyula, Budapest

XVIII., Nagyenyed u. 21. 1182

Kérem, hogy lapjukban szorítsanak helyet „piaci" tudósításoknak is, érdekesebb, színesebb híreknek külföldi gépújdonságokról (ajánlom megtekintésre a Haditechnikai Szemle egy számát mintának), mert különben kénytelen leszek visszamondani előfizetésemet. Unalmasak a hexadecimális listákkal kitöltött 10 oldalak! Akinek ez kell, vegye a Magyar Elektronikát, vagy Önök adják ki azokat külön mellékletben, de én még az első szám alapján szavaztam bizalmat a lapnak, és ha nem változik a tartalma (mint pl. volt a bécsi tudósítás. — Mint írták, megjelenésekor már elavult — de ki jár havonta Bécsbe?) én bizony lemondom az előfizetést

Már nagyon sokszor leírtam, hogy a µM olvasóinak az érdeklődése igen széles körű. Ahogyan Ön elveri rajtunk a port a hexadecimális listák miatt, mások a túl sok szöveg miatt panaszkodnak. Annak persze örülök, hogy az ausztriai piacról szóló cikkem tetszett, de az is igaz, hogy sokaknak, főleg

megszállott programozóknak nem, mert elvettem a helyet néhány programötlet elől. Új kiadónkkal megkíséreljük a lap eddigi 3-4 hónapos átfutási idejét lecsökkenteni, ez a feltétele annak ugyanis, hogy érvényes piaci információkat tudjunk közölni.

Nagy László, Inánes,

Széchenyi út 13. 3851

Kb. 1 éve olvasom lapjukat és ezalatt egyre kevesebb a ZX—Spectrumra írt programok száma. Jó volna ha ebből is annyi volna, amennyit a C64-esre közölnek.
Második kérdésem, hogy hol lehet ZX—Spectrum-ot kapni? És miben különbözik a 48 kbájtos Spectrumtól? És végül, hol lehetne megvásárolni az Ada-Winter Péter—Ada-Winter Dávid: A ZX—Spectrum című könyvet? Ha volna, részemre utánvéttellel elküldeni szíveskedjenek.
Valóban kevesebb Sinclair ZX, illetve Spectrumra írt programot készítenek és küldenek szerzőink, így mi is kevesebbet közlünk. Sinclair gépeket legjobb tudomásom szerint magánimportból lehet vásárolni a bizományosi boltokból (Bizományi Áruház, Foto-Optik, Ramovill, Skála-Metro stb.) Az Ada-Winter könyv még nem kapható. Amint megjelenik, megrendelhető a Műszaki Könyváruházban, Bp. V., Liszt F. tér 9.

Dr. Jankó Domonkos,

Közlekedéstudományi Intézet

Örömmel olvastam a Mikroszámítógép Magazin f. évi 5. számában Laczka Miklós „Mikroszámítógépek a tavaszi fesztiválon" c. cikkében KRESZ-elsősegély és munkavédelmi vetélkedő programunk kedvező értékelését, mivel a program tartalmi, szakmai részét, tehát a KRESZ-kérdéseket én készítettem és a lemezek forgalmazását is én irányítom.
Szakmai lap lévén, szeretném azonban, ha helyesbítést tennének közzé, mivel a software-t és az arra épülő programokat Dobos Imre és software-teamje (Bodnár Tibor, Dobos Katalin és Kovács József) fejlesztette ki, ill. készítette.
A „munkahelyi vetélkedő" c. lemezt egyébként továbbra is a Közlekedéstudományi Intézet forgalmazza.
Szívesen közöljük a helyesbítést.

Hargitai Tihamér, Budapest,

Napfény u. 16. 1039

Commodore 16-os számítógépet kapott a fiam ez év elején. A szükséges könyvek, folyóiratok és a témával foglalkozó hetilapok segítségével autodidakta módon eljutott egy szintre, oly annyira, hogy egy egyszerű játékprogramot is meg tud már írni. Nos ezek után körülnéztem, hogy lehetne továbbfejlődni azoknak a gyerekeknek, akik már nem kezdők?
Természetesen a saját erőből való továbbfejlődés lehetősége továbbra is megvan, de hogy egy példával éljek, egy tornásznak készülő gyermek első időben, amíg izomfejlesztés a cél, megítélésem szerint nem igényel különösebb irányítást, de amikor már megfelelő izmokkal és képességgel rendelkezik, nem hiszem hogy ugyanolyan eredményt ér el az a gyerek, akivel célirányosan szakember foglalkozik, mint akivel nem.
A számítástechnikára is vonatkozik, hogy ahová saját erőből két év alatt jut el egy gyermek, oda jó irányítással fél év alatt eljuthat.
Itt kéne találkozni az általánosan hangoztatott elveknek és a gyakorlatnak, de ezt az utat én még keresem.
Azt hiszem a magyar oktatásügy is. A fejlődés jó iskolájának tartom a μ klubokat (már amelyiket), és ha megindul az informatikai távoktatás, akkor az ott elhangzó előadások mindenki számára biztosítják a továbbtanulás lehetőségét.

Bordás Bence, Tiszaföldvár,

Rákóczi u. 2. 5430

Bordás Bencének hívnak, gimnáziumban tanulok Tiszaföldvár Cityben. Mivel eddigi próbálkozásaim sikertelenül jártak, és a lapot 1984 óta járatva (őszintén szólva akkor többet értettem volna egy kínai regényből), úgy tapasztaltam, hogy sokak problémáját sikerült megoldaniuk, én is elszántam magam, hogy leporolom az írógépet, és betöltöm magamba a gépelő programot. Nos, run.
Mivel a VC 20 ún. számítógépre nem sok játékprogramot ismerek, nem is hiszem, hogy nagyon sok lenne, elhatároztam, hogy minden BASIC tudásomat és rutinomat latba vetve, csinálok már egy „normális" játékot.
Nézzük. Végy egy jó, 3,5 kbyte-os ötletet! Van. Végy egy marék sprite-et! Hoppá, ezen a gépen egy megveszekedett fia sprite-et nem találtam, nem hogy egy marékkal. Nem baj, van definiálható karakter. Kreálj jó hátteret! Ready. Bejelentkezés, hanghatások, pontszámtáblázat. Nem téma. Lássuk, mi van még? Scroll négy irányba. Itt esik a fene . . . Lefelé, — kész van. (Kiműtöttem másfél órás, komplikált, de sikeres műtét során egy másik játékból.) Felfelé, — kész. (Print utolsó pozícióba való állítása.) Jobbra és balra. Ez az én gondom. Megkértem az iskolánk egyetlen (krónikus időhiányban szenvedő) számítástechnikához értő tanárát, segítsen. Sajnos csak elméleti tanácsokat tudott adni (amiket már tudtam), mivel a híres, nevezetes HT—1080Z számítógépben szunnyadó Z80 processzor lelkének bökdösésséhez van gyakorlata (pokeoláshoz). Sebaj, van egy barátom, aki BASIC-ből már 3 danos, de gépi kódból még csak sárga öves. Ő is nekilátott, hiszen ő tudja, hol kell bökdösni a 6531-est, hogy elérjünk nála valamit. Végén őneki sem sikerült. Nagyon örülnék, ha küldenének nekem egy ilyen gépi kódú rutint. Előre nem köszönöm meg, mert ezt a gesztust nem tartom az értelmes ember sajátjának, másrész ez „thank without program" hibajelzéshez vezethet. Szóval nagyon örülnék, ha elküldenék nekem a rutint, és ha igényt tartanak rá, szívesen elküldeném a kész programot.
Más. Mióta jár a lap, nagyon sok érdekes, felhasználható cikket, programot olvastam benne, nagyon meg vagyok vele elégedve. Tény, hogy volt egy-két gyengébb szám is, de általában minden számban található cikk vagy érthető program mindenki számára. Mióta olvasom a lapot, egyre többet értek meg belőle, és most már ott tartok, hogy most már nem lapozok el egyetlen oldalt sem azzal, hogy „ezt előbb fordítsák le magyarra". Nem értek egyet azokkal, akik kritizálják a lapban található hirdetések mennyiségét. Ez hozzátartozik a magazinszerű újsághoz. Egyébként biztos elfogynának az ezt panaszoló levek, ha az újság 60 forintba kerülne, (akkor viszont elfogynának azoknak a levelei is, akik nem tudják megszerezni a lapot).
Örömmel vettem, hogy az újság most már havonta jelenik meg. Drukkolok, hogy a sűrűbb megjelenés dacára a magazin ugyanolyan tartalmas tudjon maradni, mint ahogy azt eddig szerencsére megszoktam. End. Ready.
Ehhez a levélhez nem szabad semmit sem hozzáfűzni, így elköszön a rovat vezetője

KOVÁCS GYŐZŐ

Helyreigazítás

A μM 1986. augusztusi számában közölt Commodore 64 segédtáblázatában sajnálatos hiba miatt az alábbi javítások szükségesek:

— A **D7** mezőben DCP zp helyett helyesen **DCP zp,x** áll.
— A cikkben külön felsorolt — új — utasítások **kék színnel** jelzettek.
— A vízszintes vonallal jelölt utasításokat a processzor nem tudja értelmezni.

A téves színjelölések sajnos nehezítik, de nem gátolják a táblázat használatát. A hibákért olvasóink szíves elnézését kéri a szerkesztőség.

Kedves leendő Szerzőink!

Kérjük, hogy cikkeiket az alábbi kívánalmaknak megfelelően küldjék be:
— 2 példányban gépelve,
— a programlisták jól olvasható (nem halvány, elmosódott), normál szélességű karakterekkel, legfeljebb 40 kar/sor formátumban legyenek kinyomtatva,
— az esetleges ábrák feliratai egyértelműek legyenek.

A szerzőtől az alábbi adatokat kérjük:
— **név,**
— **személyi szám,**
— **lakáscím,**
— **munkahelyének neve, címe.**

A szerkesztőség

SZÁMALK az ORGTECHNIK

F/60-as standján minden kedves látogatóját szeretettel várja!

SZÁMALK – ORGTECHNIK

Termelésirányítási és vállalatirányítási rendszerek
Mikrogépes rendszerek, alkalmazások
Számítástechnikai alkatrészek, segédanyagok
Oktatás – Szakkönyv

SZÁMALK

SZÁMALK ORGTECHNIK

Csak két – párosításban igen merész – példa a

SOFTinvest

SZOFTVERKERESKEDELMI ÉS FEJLESZTÉSI BETÉTI TÁRSULÁS

által forgalmazott IBM PC és azzal kompatibilis számítógépeken alkalmazható szoftverekre.

3 dimenziós építész tervező rendszer:
A Nyugat-Európába már 1984 óta exportált 3 dimenziós speciális építészeti tervező program által biztosított lehetőségek közül csak néhány:

- Alaprajz interaktív tervezése képernyőn grafikus tablettel vagy „mouse"-zal
- Automatikus méretezés, terület- és térfogatszámítás
- Magasságadatok numerikus bevitele
- Homlokzati, metszeti és perspektív nézetek automatikus szerkesztése
- Kiviteli terv szintű rajzok készítése plotterre tetszőleges méretarányban
- Konszignációs listák és költségkalkuláció automatikus nyomtatása

Ára: 400 000,– Ft

VARYNFO
interaktív adatnyilvántartó rendszer
Egy program – több tucat probléma megoldása:

- személyzeti és munkaügyi nyilvántartás
- raktárkészlet, rendelések, számlák nyilvántartása
- könyvtári alkalmazások
- kérdőívek, felmérések feldolgozása
- telefonkönyv, üzletfelek nyilvántartása
- vezetői információk

Használata számítástechnikai előismeretet nem igényel, kezelése a legegyszerűbb kérdésekre adott legegyszerűbb válaszokkal történik.

Ára: 39 000,– Ft

Új címünkön várjuk érdeklődésüket:
1137 Budapest XIII., Kun Béla rakpart 8. (Újpesti rakpart) Telefon: 129-230

Nemcsak **adat, szöveg,**
hanem **grafikák,**
grafikonok
képi megjelenítésre is kiválóan alkalmas
a **PRT–80 GS**
grafikus mátrixnyomtató.

Sokoldalúan felhasználható, asztali kivitelű, elsősorban személyi számítógépek, mikro- és minigépek, intelligens terminálok, vezérlő- és mérőkészülékek adatkiviteli perifériája.
Főbb alkalmazási jellemzői:
- kétirányú nyomtatás, logikai kereséssel,
- mikroprocesszoros vezérlés,
- grafikus karakterkészlet,
- karakterkiemelés,
- normál, dőlt és dupla széles karakterek,
- szövegtömörítési lehetőség,
- automatikus soremelés,
- változtatható papírszélesség, formátumhossz, sorszélesség és sorköz,
- traktoros vagy gumigörgős papírtovábbítás,
- 80/132 kar/s sebesség.

Gyártó: BHG Híradástechnikai Vállalat
1119 Budapest, Fehérvári út 70.

BHG

Forgalmazza:
Elektromodul
Magyar Elektronikai Alkatrészkereskedelmi Vállalat
1390 Budapest, Victor Hugo u. 13–15.
Megvásárolható: Elektromodul II. sz. szakboltjában
1137 Budapest, Jászai Mari tér 5.

Fiatalok! Figyelem!

Indul az 1986. évi 8., szeptemberi számunkban meghirdetett rejtvénypályázat!

Feladatok:
1. Milyen célt szolgál az alábbi program?

```
10 DEFDBLT,S,X
20 INPUTX:X=X*0.0174532925
30 N=1:T1=X:S1=T1
40 T2=-T1*X[2/(N+1)/(N+2)
50 S2=S1+T2
60 IFABS (S2—S1)
   <1E—12THENPRINTS2:END
70 T1=T2:S1=S2:N=N+2:GOTO40
```

2. Készíts programot, amellyel a bérkifizetéskor a szükséges címletek meghatározhatók (a „címletezés" a lehető legkevesebb bankjegyet, ill. érmét eredményezze)! A program változtatás nélkül futtatható legyen a két iskolaszámítógépen, a HT—1080Z-n és a C16-on.

Beküldési határidő:
1986. október 20.

A megoldásokat írásban kérjük, szerkesztőségünk címére:
Mikroszámítógép Magazin
Szerkesztősége,
Bp. Fő u. 68.
IV. em. 452. 1027

A borítékra írjátok rá:
DIGITÁL rejtvénypályázat.
Sok sikert kívánunk!

A szerkesztőség

Barátunk, a számítógép

Beszéljünk a feladatról!

(Mottó: Mit ér, ha az ember egyik lábával a tudomány legmagasabb fokára hágott, ha közben a másik beleragadt a sárba. — Konfucius)

Kis barátunk ismertetését mindig úgy kezdik, hogy — felboncolják. Lám, ez is van benne, meg az is! Remélem, hogy az orvosudományi egyetemen a leendő orvost először megtanítják arra, hogy mi az ember. Majd a sokadik évfolyamon boncolnak csak.

Elterjedt az a tévhit, hogy az „információs rendszer", egyáltalán a rendszer, a számítógép függvénye. Mostanában napvilágot látott tankönyvünkben is a rendszert programok együttesének tekintik. A rendszertervezés = programkészítés. Ennél már Pistike is többet tud, mióta anyu receptjeit számítógépen tárolja.

Válasszuk végre szét a hobbit és a szakmát. — Hogy miért másznak? Hiszen ott van a hegy! — mondják a hegymászók. Sok programozó is úgy véli, hogy azért kell programozni, mert ott a gép. Én másként vélekedem. Ezért — viszszaélve türelmükkel — e cikkben az alapokhoz megyek visszsza. Ha unalmasnak is tűnik, ajánlom, hogy először ezt a hegyet másszuk meg, hogy utána edzetten és könnyedén jussunk fel dombokra.

Az információról

Kora reggel mondta a rádió, hogy a múlt évben az x. y. téeszben a tengeri átlagos hozama 60 mázsa volt hektáronként. Mi tagadás, ennek hallatára nem vágtam meg magam a borotvával. Már elfeledtem az egykor tanult gazdaságföldrajzból az átlagos hozamokat.

Ugyanebben a pillanatban valahol Józsi bácsi levette a pipát szép sárga agyarairól, és ennyit mondott: No! Többet nem szólt, hanem várt. (Hiába). Józsi bácsi ugyanis nem nyilvánít véleményt addig, amíg nem tudja, hogy arrafelé milyenek a földek, mekkora volt a belvízkár, miként alkalmazták a „meliorizációt". (Az utóbbi kifejezés többnyire a trágyázás irodalmi kifejezése a rádiós rovatokban, de erről a tehén nem tud.)

Az információ pontosan olyan, mint a sült csirke. A delikvens elkészült, és előre tudom, hogy erre a véznácska jószágra nagyobbik lányom azt mondja, hogy zsíros. A papa majd kijelenti, hogy sótlan, miközben én dörmögöm: Drágám, ezt picit elsóztad. A kicsi szerint a csirke égett, szerintem lehetne ropogósabb is. Ha kívülálló hallaná az elhunyt ilyen méltatását, azt hihetné, hogy itt mindenki mást eszik.

Ez a kívülálló nem számítástechnikai szakember. Nem tudja, hogy a csirke az csak „adat", és megevésével születik az „információ". (Pontosabban már színével is illatával is, de e tényezők is, az ízhez hasonlóan, eltérő benyomásokat keltenek bennünk.) E ponton az olvasó kimehet harapni valamit a konyhába, ha van utána söre. Folytathatom?

Szóval azt akarom mondani, hogy az információ legszemélyesebb tulajdonunk. Bennünk és csakis bennünk születik abban a pillanatban, amikor egy hír, egy közlés, egy új ismeret elér hozzánk, azt érzékeljük és felfogjuk. Illetve ilyenkor még nincs szó információról. Csak akkor, amikor a hírt értelmezzük. Azaz összevetjük a vele érkező egyéb hírekkel, vagy — ami technikailag ugyanaz —, a már bennünk létező ismeretekkel.

Mivel más a hátterem, mint Józsi bácsinak, a téesz-híradó nekem mást mond, mint neki. A közlésből mindenkinél többet tud meg a szomszédos téesz vezetője, aki ismeri a körülményeket. Ha pedig a nyilatkozó is hallgatja a rádiót, akkor legfeljebb saját kellemes baritonjában gyönyörködhet, mert információt nem kap. Hiszen ő adta.

Tévesek azok a vélemények, melyek szerint az adat forma, az információ pedig a forma köntösébe öltözött tartalom. Ne lenne az elfogyaszthatatlan sült csirkének tartalma? De mennyire van! Az adat a valóság valamilyen jelenségének tükörképe, míg az információ: értelmezett adat. Mivel mindenki másként értelmez, az információ valóban személyes jellegű. Én ne tudnám, aki őt főnöknek adta mindig ugyanazokat a (tartalmas) adatokat — mindig más következménnyel?

Az adatfeldolgozásról

Régi, értetlen tankönyvekben olvastuk, hogy van adatfeldolgozási célú és műszakitudományos számítógép. Folyosón hallottam: „Ezek azt a nagy böhöm gépet csak adatfeldolgozásra használják." Idéahol apa közölhette a főszakáccsal, hogy három és fél csipet majoránnával kéri a sült csirkét, amely kizárólag zsályán és nem halliszten nevelkedhetett. Volt az asztalon só, bors, paprika, és kicsit lehetett korrigálni az ízeken, de nem sokat. Hiszen már a kiindulás-

Az adat és az információ lényege

zet vezetői beszámolóból: „Elvtársak, mi nem adatfeldolgozásra, hanem vezetői információk előállítására fogjuk használni a számítógépet".

Osztán miből lesz a vezetői információ? Vagy tekinthetjük-e értelmes és a dolgokat a szó valódi értelmében „értelmezni" tudó lénynek a számítógépet? Tetszik vagy sem, a számítógépek semmi másra nem valók, mint adatfeldolgozásra. Ha majd eleget dolgozott kis barátunk, és közli velem a részben emésztett adatokat, akkor majd én szülök információt. Ezt a jogot fenntartom az emberiségnek!

Nem kívánok itt szőrszálakat hasogatni. Hiszen jómagam is meglehetősen pongyolán alkalmazom — szinonimaként — az adat és az információ megjelöléseket. Időnként azonban gondolok a lényegre is. Mert ha ezt nem tenném, akkor reményem se lehetne arra, hogy valaha is jó számítástechnikai szakember válik belőlem. A most BASIC-et tanuló Pistinek ezt a két dolgot hamar meg kellene ismernie!

Pisti még nem járt mostanában olyan magyar étteremben, ra sincs befolyása a fogyasztónak.

Aki érteni akar szakmánkhoz, annak tudnia kell, hogy vannak ún. alapadatok. Jé, ez a formás lény előttem hölgy. 99 forint a paradicsom ára, vagy annak legnagyobb közös többszöröse — kell? Az az autó piros. E mondatokban alapadatokat közöltem.

Vannak különböző fokokig feldolgozott adatok. Egyszerűbb adat az egy személy átlagjövedelmét mutató, mint a vállalati átlagjövedelmet tükröző. Így van ez a csirke esetében is: bontott csirkét, bontott csirkét . . .

Az adat egy bizonyos fokon érik meg a fogyasztásra, válik képessé az információgenerálásra. Kedvenc, saját zsargonomat kell ennek kapcsán bemutatnom.

A feldolgozási lánc

Van ilyen. Azt a szegény csirkét etették, leöldösték, felkészítették, hűtötték addig, amíg nejem tovább boncolta, és most kétségek dúlnak bennem: nejem finom fűszerezését dicsérjem-e vagy az alapvető

halízt szidjam. Milyen jó lenne, ha a folyamat egységes és általam meghatározható lenne . . .!

Van adatfeldolgozási lánc is. Ismerek egy vállalatot, ahol kedves hölgyek kiszámítják a táppénzt, a levonásokat, a pótlékokat stb. Majd az apróra darált „információt" beadagolják barátunknak, és az okos masina jókedvében még a bérlistát is kiírja. Mint a mosógépem. Fél nappal a mosás előtt beáztatom a zoknit. Majd jól kidörzsölöm a gyermekeim élénkségét tanúsító nyomokat. Zutty bele a mosógépbe! Egy óra múltán vegyesnek mondott színekben kapok vissza pár zoknit, amit már csak napokig kell szárítani az automatikus centrifugálás után.

Tanfolyami szünetben mondja mérnök ismerősöm (számítógépet gyártó vállalat alkalmazottja) kérdésemre, hogy mit tesznek a kimenettel (vagyis outputtal): — Egy alkalmatos targoncán az üzem olyan sarkába toljuk, ahol nem zavar. A nevezett lista tájékoztat arról, hogy a múlt dekádban melyik gép volt lerobbanva, milyen következménnyel. Mérnök barátom pedig beadagolhatja a gépbe azt, hogy valójában(!) mi történt a múlt dekádban. Elvégre az egyéb kimutatásoknak konzisztenseknek kell lenniük. Miért nem lehet a változás pillanatában új termelési programot adni a gépnek? Vagy ilyenre javaslatot kérni tőle?

Sokan úgy vélik, hogy a számítógép is a közismert reklámnak megfelelően viselkedik:

„A mosógép akkor érzi jól magát és akkor takarékos, ha jól megtömik"

Nem vagyok naiv. Ismerem a gépi korlátokat. Szentül meg vagyok győződve arról, hogy a feldolgozási lánc elején is és végén is elkényeztetjük barátunkat, a számítógépet, mert lusták, tehetetlenek, pontatlanok, megbízhatatlanok vagyunk. És ezért gyors barátunk helyett kétszer, tehetetlenül, pontatlanul és megbízhatatlanul kell előre felkészítenünk és utólag kiigazítanunk pontos kis barátunk „információit". És bérelszámoló hölgyek, termelésprogramozó urak szidják barátunkat, aki munkát ad.

No, a nagygépek kisöccse, a személyi számítógép végre megérkezett! Aki picit is komolyan játszik vele, annak szeméről lehull a fátyol. Kollégáim, reszkessetek! Ha nem tartjuk távol a mikrogépektől a felhasználót — amiért sokat megteszünk —, akkor kiderül: nem vagyunk már a távoli számítógépisten titokzatos papjai, akik nehezen érthető áldozatokra kényszeríthetik a felhasználót. — Jé, hiszen itt valóban alapadatot lehet bevinni, és egy kicsit tovább lehet feldolgozni, mint azt nekem mondták, mindenféle ürügyekre hivatkozva — szól majd Kovács egy napon. Kovács, a végső felhasználó.

Azon a napon Szabó, a rendszerfejlesztő, rájön arra, hogy miközben a programozás magaslataira hágott, a másik csizmája egyre terhesebb lett. Keményen kell kihúzni. Keményen kell lépni annak érdekében, hogy kényelmünket feledve, ellássuk feladatunkat. Amelynek lényegét már Pistike is érzi: feladatunk a szolgálat. A kényelemteremtés.

Ennyit az információkról és a sült csirkéről.

DR. HALASSY BÉLA

data manager

MINDEN EGYSZERŰ, HA VAN EGY JÓ MENEDZSERE!

Aki:
- **gazdag piaci és szakmai ismerettel rendelkezik,**
- **ismeri a korszerű gépeket és alkalmazásukat,**
- **ért a szervezéshez és a műszaki fejlesztéshez,**
- **gyakorlott programozó,**
- **széles választékból rövid határidőre szállít,**
- **pontosan tudja a feladatát és nagy rutinnal irányítja a többiek munkáját.**

ÖNNEK CSAK DÖNTENI KELL, A TÖBBIT ELVÉGZI A

data manager
SZÁMÍTÁSTECHNIKAI KISSZÖVETKEZET
1134 Budapest Dózsa György út 150
Postacím: 1553 Budapest Pf. 41
Tel.: 202-650/247
Tx.: 22-6741

86 orgtechnik hungaria budapest'86 E/52 stand

CONTROLL

KOMPLEX SZÁMÍTÁSTECHNIKAI VÁLLALKOZÁS

Kulcsrakész ügyviteli rendszerek a **MICROCONTROLL** számítógépcsalád elemeivel

MICROCONTROLL '86–'87 számítógépcsalád

Teljes IBM XT és AT hardver-szoftver kompatibilitás
Több munkahelyes és hálózatos rendszerek kialakítása
Széles körű perifériaválaszték
Grafikus és NYÁK-tervező rendszerek
Mágnesszalagegység és videomagnó-illesztés
COMMODORE floppy illesztés

MICROCONTROLL '84 számítógépcsalád

CP/M kompatibilis operációs rendszer
Nagy teljesítményű lokális hálózat
Kimagasló adatfeldolgozási sebesség
Igen kedvező teljesítmény-ár viszony.

INTELLIGENS ADAT-ELŐKÉSZÍTŐ RENDSZER

Tetszőleges számú munkahellyel
Kezeléstechnikai kompatibilitás
ismertebb rendszerekkel

KOMPLETT
FELHASZNÁLÓI RENDSZEREK

Egy- és több munkahelyes
felhasználói rendszerek
Szervezés, programozás
Betanítás, adaptálás

PROFESSZIONÁLIS SZÁMÍTÁSTECHNIKAI SZAKÜZLET

1132 Bp., Visegrádi u. 6. Tel.: 128-064
Szaktanácsadás
Adás-vétel
Lízing

CONTROLL

Számítógépeinket 1-3 hónapon belül szállítjuk
Gyors, megbízható szerviz, installálás
Oktatás, szervezés

Elektronikai és Számítástechnikai Kisszövetkezet
1027 Bp., Szász Károly u. 2. Tel.: 158-428.

86 orgtechnik hungaria budapest'86

ÁPISZ KERESKEDELMI VÁLLALAT

a B/9 standon

a számítástechnikában hagyományos és már megszokott cikkein kívül

bemutatja **ÚJDONSÁGAIT**

TEDI AUTOMATIKUS TELEFONHÍVÓ ÉS HÍVÓSZÁMTÁROLÓ KÉSZÜLÉKÉT

helyi, belföldi és nemzetközi távhívásokat automatikusan bonyolít, 40 hívószámot tárol, hálózatkimaradás esetére akkumulátoros memóriavédelemmel van ellátva.

IBM kompatibilis mikroszámítógépét különböző felhasználásokra, teljes programellátottsággal

IRODAI HORDOZHATÓ SOKSZOROSÍTÓGÉPÉT

A/4 méretű
4 szín ellátottságú
programozható
folyékony kristály
kijelzésű

SZEMÉLYI SZÁMÍTÓGÉPEK PROGRAMJAIT ÉS TARTOZÉKAIT

fényceruza
botkormány
alkalmazási programok
játékprogramok
perifériaillesztők
memóriabővítők

MINDEZEKET A HELYSZÍNEN ÁRUSÍTJUK, illetve előjegyzést veszünk fel

Minta utáni árusítás

BITEK ÉS FIGURÁK

Állásértékelés V.

Figuratámadás és -védelem

Az értékelőfüggvénynek eddig ismertetett komponensei a sakkjáték stratégiai szempontjait ölelték fel. Ezek a stratégiai szempontok biztosítják az állás felmérését és képessé teszik a programot az állásban lezajló folyamatok lényegi megértésére. A figurák támadásának és védelmének figyelembevételével a programunkat valamelyest taktikailag is felvértezzük, és így reálisabban értékeli az állás dinamikáját és taktikai lehetőségét. Mint a későbbiekben látni fogjuk, ez a taktikai játékhoz önmagában még nem elegendő, de a hozzá vezető lépcső első foka.

A figurák támadása és védelme — mint kvantitatív mennyiség — meghatározza az egyik félnek az ellenség figuráira kifejtett támadási esélyeit és a saját figurák védelmének mértékét. Tudjuk, hogy az a játékos van előnyben — figuratámadás és védelem szempontjából —, amelynek figurái jobban védettek és az ellenfél bábjait aktívabban támadja. Ez nem jelenti azt, hogy ebben az állásban az előnyben levő félnek azonnali taktikai esélyei vannak, hanem csak azt, hogy kombinációs lehetősége nagyobb. David Lavy megemlíti könyvében ezeket az értékeket is, amit az *ábrán* látható példán mutat be.

Ez az állás a francia védelem egyik változata, amely a következő lépések megtételével alakul ki:

1. e2 — e4, e7 — e6
2. d2 — d4, d7 — d5
3. Hb1 — c3, Ff8 — b4

Látható, hogy világos csak e4 gyalogjával támadja az ellenfél d5 gyalogját, így egy támadó értéke van. Itt nem kell figyelembe venni, hogy a d5 gyalogot sötét biztonságosan védi az e6 gyaloggal és a d8 vezérrel. Ez az értékelőfüggvény egy másik komponensében realizálódik. Sötét d5 gyalogjával világos e4 gyalogját támadja, b4 futójával a világos c3 huszárt. Tehát világos egy támadásával szemben sötétnek két támadó értéke van.

A figuratámadás kvantitatív meghatározása több módon történhet. Meghatározhatjuk mindkét fél esetében az ütések számát, figyelembe véve a köteléseket is. Jelen esetben nem a félig legális lépésekből adódó ütések számát kell figyelembe venni, hanem csak a legális lépésekből adódókat. Ez azért lényeges, mert nagyon sok taktikai lehetőség éppen a kötések kihasználásán alapul. Ezért ezt a programnak feltétlenül figyelembe kell vennie. Az előbbi állásban is észrevehető, hogy b4 futó leköti a c3 huszárt, és így világos e4 gyalogja védtelen marad. Sokkal pontosabb értékhez jutunk, ha a támadó figura értékét a megtámadott figura értékéhez viszonyítjuk, és figyelembe vesszük, hogy a megtámadott figura védett-e. Ezek alapján a támadás értékét a következő módon is meghatározhatjuk:

$$\frac{\text{a „megtámadott” értéke}}{\text{a támadó értéke}} \Big/ \text{védők száma} + 0{,}25$$

Ezt a képletet felhasználva az előbbi diagramon a következő figuratámadási értékeket kapjuk:

1. e4 gyalog támadása d5 mezőre: (1:1) : 2,25 = 0,4
2. d5 gyalog támadása e4 mezőre: (1:1) : 0,25 = 4,0
3. b4 futó támadása c3 mezőre: (3:3) : 1,25 + 0,8

Láthatjuk, hogy sötétnek 4 × 4 egység a figuratámadási előnye.

A figuravédelem meghatározásához a védő figura értékét is figyelembe kell venni. Ezt a legegyszerűbben a védő figurák reciprok értékének összegével állapíthatjuk meg, amiből kiderül, hogy a legelőnyösebb, ha a saját figurát a lehetséges legkisebb értékű figurával védjük meg. Ezek szerint a megtámadott figurát elsőként gyaloggal, ha nem lehet, akkor huszárral, futóval, bástyával, vezérrel, majd királlyal célszerű megvédeni. Ez azért fontos, mert ha a védelemre nagyobb értékű figurát használunk, amely egyben hatékonyabb is, akkor ez a támadásban háttérbe szorul és passzívvá válik. Ezt a módszert az előző állásra alkalmazva, a következő értékeket kapjuk:

Világos:
Fc1: 1/5 + 1/9 = 0,3
Vd1: 1/4 = 0,3
Ff1: 1/4 = 0,3
Hg1: 1/5 = 0,2
a2: 1/5 = 0,2
b2: 1/3 = 0,3
c2: 1/9 = 0,1
f2: 1/4 = 0,3
g2: 1/3 = 0,3
h2: 1/5 = 0,2
Hc3: 1/1 = 1,0
d4: 1/9 = 0,1

Sötét:
Hb8: 1/5 = 0,2
Fc8: 1/9 = 0,1
Vd8: 31/4 = 0,3
Hg8: 1/9 = 0,1
a7: 1/5 = 0,2
b7: 1/3 = 0,3
c7: 1/9 = 0,1

Az egyes figurák értékeinél tisztán látható, hogy ha a gyalogot egy egységnek vesszük, akkor a huszár és a futó 3 egység, a bástya 5 egység, a vezér 9 egység. Kérdés, a királyt hogyan értékeljük anyagi szempontból. Mivel ez kitüntetett báb a sakkjátékban, anyagi értékéről nemigen szoktunk beszélni. Jelen esetben kivételt teszünk, és a királyt is anyagi értékben részesítjük. Ezt megtehetjük dinamikusan úgy, hogy amíg a táblán levő tisztek száma egy bizonyos értéket meghalad, addig a király értékét száznak vesszük, a végjátékba történő átmenet esetén pedig néggyel számolhatunk. Ezzel elérjük, hogy a középjátékban, amikor királyunkat könnyen megtámadhatják és mattveszélybe kerülhet, ne hagyja el biztonságos helyét, hogy valamelyik figuráját megvédje. A végjátékban viszont — ahol a király centralizálása igen fontos, és nyugodtabban is mozoghat a tábla közepén bántódás nélkül — az anyagi értéket vehetjük négy gyalogegységnek. Ezzel értéke egy könnyű tisztnél nagyobb lesz, de a bástyánál kisebb, ami általában reális is.

A konkrét figuratámadás kiszámításának módjával később foglalkozunk, miután a királytámadás és királybiztonság kvantitatív meghatározását ismertettük.

KOVÁCS P. ATTILA

NEMZETKÖZI FOTÓ- ÉS RAJZPÁLYÁZAT

A Nemzetközi Távközlési Egyesület (UIT) nemzetközi fotó- és rajzpályázatot hirdet a 160 tagországban élő fiatalok számára. Hazánk részvételét a nemzetközi versenyben a Magyar Posta szervezi. A verseny általános témáját „A távközlés a fejlődésért" címmel jelölték meg a kiírók. A pályázók gondolkodását négy altémával segítik:

A műhold csökkenti a távolságokat.

A telefon a hétköznapi életben.

A televízió oktatásra is alkalmas.

Mire jó a számítógép?

A verseny résztvevői fotók, rajzok, festmények, egyéb illusztrációk (pl. kollázsok) segítségével dolgozhatják fel témáikat. A versenyben 8–18 év közötti fiatalok vehetnek részt három korcsoportban: 8–12, 13–15, 16–18.

A magyarországi pályázóknak 1986. november 30-ig kell eljuttatniuk műveiket a Magyar Posta Központja tájékoztatási és marketing osztályához, személyesen a Budapest XII. Krisztina krt. 6–8. vagy postán az 1540 Budapest címre. A Magyar Posta által összehívott nemzeti zsűri dönti el, hogy a beérkezett munkák közül melyek kapják a Magyar Posta díjait és közülük melyeket küld el a posta (kategóriánként maximum 10-et) a genfi nemzetközi versenyre.

A tagországokból Genfbe küldött pályaműveket 1987. október 20–27. között mutatják be a TELECOM '87 világkiállításon. A nemzetközi zsűri itt választja ki a három korcsoport díjazásra javasolt pályamunkáit.

1983-ban egyébként 53 ország nemzeti pályázata nyomán 300 ezer alkotás született, a nemzetközi zsűrinek pedig 1500-ból kellett választania. A magyar fiatalok rendkívül eredményesen szerepeltek, a 16–18 évesek között Budai Tamás világelső lett.

A pályázattal kapcsolatos kérdésekkel a Magyar Posta Központjának tájékoztatási és marketing osztályát kereshetik meg az érdeklődők az 555-575 vagy a 754-893 telefonszámon.

Keresünk megvételre jó állapotban lévő VIDEOTON gyártmányú VDT 52103, VPC, VPPC displayket.

Villamosenergiaipari Kutató Intézet

Ügyintéző:
Papp György osztályvezető
Telefon: 178-698

PRODUKTORG
SZERVEZÉSI VÁLLALAT

C64, M08X, IBM PC XT kompatibilis rendszerekre ajánljuk programcsomagjainkat:

— főkönyvi könyvelés,
— költségfelosztás,
— állóeszköz-nyilvántartás,
— rendelés-nyilvántartás,
— bérelszámolás,
— adóelszámolás.

1027 Budapest, Fő u. 68.
Telefon: 154-090

Bővebb felvilágosítással készséggel állunk rendelkezésükre!

Beruházók vállalkozók figyelmébe!

A fejlesztések, beruházások, fővállalkozások sikeres irányításához nélkülözhetetlen a különféle költségek alakulásának állandó nyomon követése és a költségek folyamatos ellenőrzése.
Ezt a követelményt elégíti ki

a KFR-MICRO

költségfigyelési programcsomag!

Alapvető szolgáltatások: költségek alakulásának nyomon követése, költségösszesítések készítése, százalékos előrehaladások számítása, kerettúllépés figyelése, várható bekerülési összeg számítása stb.

A KFR-MICRO a következő számítógépeken üzemeltethető: TAP–34, M08X PROPER–8, PROPER–16, COMMODORE–64, IBM PC

Tekintse meg a SYSTEM Szervezési Vállalat kiállítását a Budapest Sportcsarnokban 1986. október 3–8. között a C/25-ös standon az ORGTECHNIK kiállításon

86 orgtechnik hungaria budapest'86

ADOK–VESZEK–CSERÉLEK

■ Ebben a rovatban rövid szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közleményeknek gépelt soronként (60 karakter) 100,— Ft, magánszemélyeknek az első sor 50,— Ft, minden további sor 20,— Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

■ Atari 800 XL új személyi számítógép magnóval, német nyelvű leírásokkal, áron alul eladó. Járfás, Ajka, Márvány u. I. 8451.

■ ZX81-et keresek megvételre, vagy árának megfelelően könyvet, alkatrészt, barkács-, fotóeszközt adok érte. Lehetőleg bővítővel! Varsányi Gábor, Kiskőrös, Árpád u. 12. 6200.

■ C64-re készült programokat cserélek. Szomor László, Bp., Báthori u. 18. 1054. Telefon: 320-807.

■ Commodore 16 (módosítással C64) idegen szavakkal tetszés szerint feltölthető *nyelvtesztszótár* programkazetta utánvéttel (300 forintért) eladó vagy/és ZX81-hez turbo 81; 1 k-s autóverseny, BASIC nyelvoktató kazettákat cserélek C16 programokra. Dakó Csaba, Dunaújváros, Martinovics u. 31. I. 1.

■ Commodore 64-hez MC 381 típusú, BRG gyártmányú adatmagnó 4000 forintért eladó. Gyári garancia novemberig. Debreceni László. Telefon: 353-900/543-as mellék.

■ Laser-210 számítógép magnóval, botkormánnyal eladó vagy bérbe adó. Tanulásra kiválóan alkalmas. Bártfai Barnabás, Agárd, Széchenyi u. 74. 2484. Telefon: 239.

■ PL—150 típusú szalaglyukasztó elektronikával eladó. Várszegi Sándor, Bp., Beregszász u. 77. 1118.

■ Olcsón eladó vagy erősítőre cserélhető egy TI 99/4A 16 k-s színes számítógép programokkal. Jelinek András, Bp., Mogyoródi út 117. 1141.

■ Commodore 64+VC 1541 lemezegység+38 db lemez tele programokkal+2 db botkormány 57 ezer forintért eladó. Strelli József, Bp., Török Flóris u. 28. VI. 36. 1203. Telefon: 451-350, 8—16 óráig.

■ C64-es játékprogramokat cserélek magnót használókkal. Soós Sándor, Szeged, Gyöngyvirág u. 18/A. V. 12. 6723.

■ Eladó ZX—Spectrum 48 k — 14 000 forint, interface II. — 3000 forint, Computone magnó — 3000 forint, kb. 50 db program — 1500 forint, kb. 1500 oldal irodalom — 1500 forint. *Az egész együtt csak 22 000 forint!* Bendik István, Bp., Harmat u. 71/B. 1104. Telefon: 572-824.

■ Eladó a Mikroszámítógép Magazin 1984-ben és 1985-ben megjelent összes száma. Csatlós Béla, Mezőtúr, Mező 1. 47.

■ TI 99/4A számítógép botkormánnyal, magnóval és játékprogrammal 20 000 forintért eladó. Telefon: 62/56-607.

■ VIC—20-hoz oktató- és játékprogramok olcsón eladók. Krusó Oszkár, Tát, Néphadsereg u. 21. 2534.

■ Eladó ZX—Spectrum, 48 k, sok magyar nyelvű dokumentációval (BASIC és gépi kódú programozás, hardversegédlet, szervizkönyv), GP—50S nyomtató és sok program. Varga Zoltán, Bp., Murányi u. 36. IV. em. 1078. Telefon: 216-975.

■ ZX81 16 k-s bővítővel eladó. Id. Krasznai László, Bp., Sáfrány u. 67. 1116. Telefon: 867-745.

■ 64 kbájtos számítógép floppyval, programokkal olcsón eladó. Telefon: 410-158.

■ C64-re játék- és másolóprogramokat cserélnék. Elsősorban lemezen, de kazettán is. Lucz Géza, Kaposvár, Kinizsi ltp. 5. 7400.

■ C64-es programokat cserélek bel- és külföldi partnerrel. Tóth József, Bp., Bercsényi u. 2. 1041. Telefon: 890-226.

Nemzetközi Számítástechnika — Internacia Komputado

Az 50 országban olvasott magazin
példányonként 30 forintért kapható
az SKV Könyvesboltban.
Bp. II., Keleti Károly u. 10.
Telefon: 158-018

86

HÁLÓTERVEZÉS = HSZR-MICRO

Ismerjék meg és használják Önök is a HSZR-MICRO hálótervezési programcsomagot,

melyet eddig több mint száz hazai vállalat, illetve intézmény alkalmaz sikeresen. Referenciákkal rendelkezünk: beruházásszervezés, kivitelezésszervezés, karbantartás-szervezés, termelésirányítás, mezőgazdaság, oktatás stb. területeken. A HSZR-MICRO 8 és 16 bájtos mikroszámítógépeken üzemel a COMMODORE-tól az IBM PC AT-ig.

A HSZR-MICRO FŐBB SZOLGÁLTATÁSAI:
- **hálószerkesztés és rajzolás képernyőre, sornyomtatóra,**
- **kombinált MPM/CPM módszer,**
- **vonalas ütemterv változtatható időtengellyel,**
- **erőforrásterv, aggregáció, hisztogram,**
- **aktualizálás, nyomon követés, újraütemezés,**
- **szabadon változtatható output tablók.**

Tekintse meg a SYSTEM Szervezési Vállalat kiállítását a Budapest Sportcsarnokban 1986. október 3–8. között a C/25-ös standon az ORGTECHNIK kiállításon

PRO-KONTRA GM.
1074 Budapest,
Csengery u. 7. fszt. 1/a
☎ 417-893

Cégünk ajánlataiból:
- **64 k-s memóriabővítő Commodore C16-hoz**
- **Személyi számítógépek és összes tartozékaik garancián túli javítása átalánydíjas szerződéssel is**
- **FAST–VC–1541 kommunikációgyorsító rendszer a C64-hez (minden gép—floppy közötti és floppyn belüli műveletet 4–12-szeresére gyorsít)**

Konfiguráció beszereléssel együtt 7000,– Ft

- **Abszolút programvédelem C64-hez: cartridge-ben, műgyantával kiöntött EPROM-ba égetéssel, MÁSOLHATATLANNÁ teszi programjait**
- **IEC buszról vezérelhető mérésportváltó egység 2×30 vagy 1×60 bemeneti, 2 ill. 4 kimeneti csatornával**
- **Digitális kijelzésű elektronikus óra 8×14 cm-es digitmérettel, különféle színekben**
- **Egyéb, közepes sorozatú fejlesztések igény szerint**

Pro-Kontra Automatizálási, Műszaki Tanácsadó és Közvetítő GM

Budapest, Csengery u. 7. fsz. 1/a 1074
Tel.: 417-893
Levélcím: Budapest, Pf. 72. 1581
Telex: 22-7770

Szidarovszky Ferenc–Molnár Sándor: Játékelmélet műszaki alkalmazásokkal. Többcélú programozás, klasszikus és differenciáljátékok. (Budapest, 1986. Műszaki Kiadó, 240 oldal. Ára: 75,– Ft.)

Bonyolult rendszerek elemzése során gyakran előfordul, hogy a vizsgálati szempontok egymásnak ellentmondóak, esetleg nem összemérhetők. A konkrét feladatok jellege szerint játékelméleti, többcélú programozási módszert kell választani, vagy a differenciáljátékok valamely módszerét kell alkalmazni. A könyv ezzel a három, egymással szoros kapcsolatban álló témakörrel foglalkozik.

Minden C64-tulajdonos, programozó, felhasználó örömmel üdvözölheti a Novotrade Rt. vállalkozását, a Data Becker kiadó C64-gyel foglalkozó sorozatának folyamatos, magyar nyelvű megjelentetését. A kötetek sok kipróbált, ellenőrzött BASIC és assembly nyelvű példaprogram listáját tartalmazzák fénymásolt, nem nyomdai úton szedett formában. Az assembler lista mellett mindig megtalálható a BASIC betöltőprogram is. Az így közölt programrészletek, programok garantáltan jók, hibátlanok. Ugyanez nem mondható el a szöveg közben magyarázatként szereplő néhány soros programokról, parancsokról, utasításokról. Ezek között több sajtóhibát találtam. Tanár vagyok, így az ismertetések sorrendjét egyben didaktikai sorrendnek is ajánlom. Az egyes könyvek átfedései elkerülhetőek lettek volna ugyan, de sosem árt a téma magasabb szinten való, más szempontú ismétlése, újratárgyalása. Az árakról nem beszélek.

Angerhausen — Englisch — Gerits — Schellenberger: A BASIC programozás magasiskolája C64-esen (Budapest, 1985. DATA BECKER — NOVOTRADE RT, 183 oldal. Ára: 308,— Ft.)

Az első fejezet a programozáselmélet és az ösztönös programkészítés összevetése gyakorlati problémák tükrében. Nem mondja ki ugyan, de világosan szemlélteti, hogy a programkészítés lényegesen szélesebb, általánosabb fogalom, mint a kódolás fogalma. Az ötlet születésétől a feladat precíz megfogalmazásán keresztül, az algoritmus-készítésen át a változók megválasztásával, a képernyők, a számítás, az adatfájlok, a nyomtatási képek tervezésével, a folyamatos és részletes dokumentálás szükségességével foglalkozik. Az egyes részek tárgyalásakor sok hasznos, többször alternatív megoldási lehetőséget, segédeszközt ajánl, mutat be.

A második fejezet a program részeivel foglalkozik a klasszikus felosztás — bevitel, feldolgozás, kivitel — csoportosításban. Ezekkel kapcsolatban olyan kisebb-nagyobb, de rendkívül általános és hasznos tudnivalót, eljárást közöl, melyek nélkül nem képzelhető el egy jó program. A bevitel tárgyalása közben a szerzők megadnak egy általános, egységes „bolondbiztos" és barátságos képernyőmaszk-készítő és adatbeolvasó rutint, melyet mindenki beépíthet(ne) saját fejlesztésű programjába. A feldolgozás tárgyalásakor a változók helyes típusának megválasztása és a számítási pontosság problémája mellett felhívják az olvasó figyelmét a különböző rendezési eljárások hely- és időigényére. Ismertetik a buborék- és gyorsrendezést. Hiányolom itt legalább néhány keresési, kiválasztási algoritmus, illetve a memóriában való rendezések szélesebb körének ismertetését, nem beszélve a háttértárolón őrzött adatok összefuttatásáról.

Az adatok kiírásával foglalkozó rész összevonja az eredmények nyomtatón való megjelenítésének és a lemezen történő tárolásának problémáit. Ismerteti a VC 1541-es lemezegység SEQ és REL típusú fájljainak műveleteit, majd sajnálkozva közli, hogy a sokkal hasznosabb ISAM fájl megvalósítását az operációs rendszer nem támogatja. Ezt követi a könyv leglényegesebb része, a szerzők által létrehozott QUISAM fájl részletes ismertetése a hozzá szükséges HASH kód leírásával együtt. Végül a nyomtató kerül sorra. Az összegzés mottója: a felhasználót az eredmények érdeklik! Egy általánosan használható, a programozó által paraméterezhető nyomtató rutin zárja a fejezetet.

A harmadik fejezet a programozástechnikai kérdések tárháza, bár itt fejeződik be az első fejezet gondolatmenete is. Tárgyalja az évelő (általános problémakörre készített) programok problematikáját: a paraméterezést. Ismerteti a menütechnika és funkcióbillentyűk használatának előnyeit, mindezt a programok arculatának egységesítése miatt. Visszatérve a programozás lépéseire, röviden vázolja a „kész" programok tesztelésének szükségességét, tanácsokat adva a lehetséges segédeszközökre, bővítésekre vonatkozóan. Sajnos a különböző tesztelési módszerek ismertetésére a szerzők nem térnek ki. Végül hangsúlyozzák a rövid, világos, közérthető kezelési utasítás szükségességét. Itt-ott megemlítik ugyan a programlista olvasható voltának, a program fejleszthetőségének fontosságát, azonban ennek tárgyalására nem vállalkoznak. A fejezetet az építkező, (alulról felfelé) és az elemző (felülről lefelé) programozási módszerek megemlítésével, valamint három memóriatakarékossági eljárás megoldásával zárják.

A negyedik fejezet az alkalmazásoké. A szerzők konkrét példákon keresztül mutatják be az eddig ismertetett elvek megvalósításait. Tárgyalnak egy paraméterezett raktárnyilvántartást, egy általános soros fájl listázót, egy egyszerű szövegszerkesztőt, a QUISAM fájl alkalmazását irodalomjegyzék nyilvántartására, valamint egy ismerőseink címnyilvántartását végző programot.

Az ötödik fejezet a programkészítéshez felhasználható segédprogramok egy részét foglalja össze. Ismerteti a MASTER 64 és a STRUKTO 64 programokat.

ÉNEKES FERENC

Beruházók figyelem!

A beruházások pénzügyi nyilvántartásához COMMODORE 64 számítógépre ajánljuk

86 orgtechnik hungaria budapest'86

a számla-nyilvántartási, forráskezelési és tőkésgép-nyilvántartási programcsomagokat.
Alapvető szolgáltatások:
a kötelező nyilvántartások elvégzése,
adatszolgáltatás a jelentésekhez,
információ a beruházások irányításához.

Beruházásszervezést, komplett beruházásirányítási rendszer bevezetését is vállaljuk.

Tekintse meg a SYSTEM Szervezési Vállalat kiállítását a Budapest Sportcsarnokban 1986. október 3—8. között a C/25-ös standon

az ORGTECHNIK kiállításon

86 orgtechnik hungaria budapest'86

SALDO

PÉNZÜGYI SZERVEZŐ ÉS TANÁCSADÓ VÁLLALAT

több évtizedes tevékenységei – szervezés – tanácsadás mellett újabb szolgáltatásokkal segíti a hatékony vállalati gazdálkodást

- értékelemzés
- gazdasági elemzés
- könyvszakértői tevékenység
- lízing tevékenység

Szervezési tevékenységünk alapja a hazai és külföldi elektronikus adatfeldolgozó berendezésekre, kisszámítógépekre kidolgozott programcsomagok, egyedi szervezések.

A hagyományos ügyviteli gépek (R. 1355, R. 1711, R. 1720, R. 5100-as gépcsalád) mellett profilunkba illesztettük a hazai piacon megjelent géptípusok közül a következőket:

- COMMODORE 64
- ROSY 80F
- VT 16
- R. 1715
- IBM PC XT
- ATARI 800XL

Programcsomagjaink, mintaszervezéseink az ügyviteli folyamatok teljes körét felölelik, elsősorban az ipar, mezőgazdaság, költségvetési ágazatok területén. Szakmai tanácsadással, konzultációs lehetőségekkel, bemutatókkal segítjük a helyes gépkiválasztást. Megbízóink igényére egyedi szervezési munkákat is vállalunk a felsorolt géptípusokra. Szoftvereink bevezetését folyamatosan szervezett oktatással biztosítjuk.

Rendszeres felvilágosítás:

Bemutatóterem
Bp. VI., Paulay Ede u. 15.
Tel.: 421-764, 220-607

HAVI AJÁNLATUNK:

- EPSON illesztőkártya
- Winchester illesztőkártya
- 14,5 Mbájtos Winchester
- olcsó hajlékony lemez
- Streamer (backup)
- mángnesszalagok

VÁRJUK ÖNT:

hétfő	
kedd	9—16 óráig
szerda	
csütörtök	9—17 óráig
péntek	9—15 óráig

LEGYEN A PARTNERÜNK!

Keresse fel bizalommal a Műszaki Árut Értékesítő Vállalat és az „5G" Számítástechnikai Szolgáltató Kisszövetkezet Számítástechnikai szakboltját!

Címe: Budapest XIII., Victor Hugo u. 33.
Tel.: 494-782

Vevőszolgálat és szaktanácsadás!
Eladás és lízing!
IBM PC/XT, IBM PC/AT
COMMODORE, ATARI
számítógépek

Menetjegyet mikrogéppel

A Nyugati pályaudvaron megjelent az országban az első nyolc mikroszámítógép, amely a MÁV-menetjegyek kiadását végzi. Ezek a célgépek Intel 8085 típusú mikroprocesszoron alapulnak, s egybeépítve tartalmazzák a megjelenítőt, a nyomtatót, a billentyűzetet és egy hajlékonylemezes tárolót is. A célorientált gépkialakítás a kezelés egyszerűsítését szolgálja.

A mikrogép alkalmazása végső soron nem gyorsítja ugyan magát a menetjegy-kiszolgálást, viszont közvetve jelentős, az utasok számára is kedvező változást eredményez. Eddig a pénztárosnál kint lévő jegyállomány igen nagy értéket képviselt. Ennek a kötelező, rendszeres leltározása vitte el a havi munkaidőalap jelentős részét. A számítógép alkalmazásakor a pénztárosok csak üres tekercseket kapnak, ami értékké csak a rányomtatás során válik. A forgalom összesítését egy gombnyomásra azonnal elvégzi a gép, amely komoly munkaidőalapot szabadít fel. Az utasok számára ez több pénztárablak egyidejű nyitva tartásával válik érezhető előnnyé.

GYSEV 106F 86.03.19
Budapest-Nyugati pu.
Kisért kutya Egy útra
ÉRVÉNYTELEN
Bármely vonat 2. oszt
Érv: 86.03.19 +1nap
0 5 0 km 8Ft 02806

Mintaként kinyomtatott kutyajegy

86.03.19 106F 011
Db. 02806 Ft. 0194964
Db. 02806 Ft. 0194964
Bevétel: Ft. 0000000
Rontott:
Db. 00000 Ft. 0000000
Terhelés: Ft. 0000000

1061 Ft. 0000000 000
1062 Ft. 0000000 011
1063 Ft. 0000000 000
1064 Ft. 0000000 ,000

Elszámolási összesítő minta a gépen dolgozó négy pénztárosra lebontva

Új házi nyomtató

VT 21 100 néven, Centronics licenc alapján új mikronyomtatót kezdett gyártani a Videoton. Az alig 3 kg súlyú mátrixnyomtató egy sorba maximum 80 karakter nyomtatására képes. Háromféle papír befogadására is lehetőséget nyújt: a szabványos A/4-es papírt gumihengerrel, a leporellót úgynevezett traktorral továbbítja, de tekercspapír-adaptere is van. Nyomtatási sebessége 90 karakter/perc, ami kategóriájában igen jó, hiszen megfelel a hazai professzionális mikronyomtatók sebességének. Külön piszkozat-tisztázat nyomtatási lehetősége is van. A mikroszámítógéphez annak mind a soros, mind a párhuzamos illesztőjén keresztül csatlakoztatható. Az új nyomtatót a Videoton már be is mutatta saját házi számítógépéhez, a TV-Computerhez csatlakoztatva. Ára: 25 ezer forint.

GENERALCALC

Háromdimenziós táblakezelőt készített az NDK-beli robotron cég. A 256 × 30 × 15 elemet kezelő program már elkészült. Jelenleg integrált szoftverré alakítását végzik, azaz kiegészítik szövegkezelői, adatbázis-kezelői és grafikus funkciókkal is. A fejlesztést az IBM PC-vel kompatibilis, robotron gyártmányú A 7100 típusú gépen végzik. Már folyik a GENERALCALC-nak nevezett termék — várhatóan 1987 elején történő — külföldi piacra hozásának előkészítése is. Ez ügyben a robotron nemcsak amerikai, hanem magyar forgalmazókkal is tárgyal.

A pénz számolva...

A pénzintézetek nagyobb hivatalaiban több százezer forint az érmeforgalom. Az ebből eredő feldolgozási munkát könnyíti meg a kiskörösi PROTOKON kisvállalkozás tőkés importot helyettesítő mikroszámítógépes érmeszámláló készüléke, mely a tavaszi BNV-n mutatkozott be először a nagyközönség előtt. A gép eddigi eladási sikerei arra a reményre jogosítanak fel, hogy a KGST és a tőkés piacon is lesz rá kereslet. A posta és az OTP után jelenleg a Magyar Nemzeti Banknál folynak a gép üzemi „nyúzópróbái".

Összefogásból számítógép

Bács-Kiskun megyében nagy hagyománya van az „Egy üzem — egy iskola" mozgalomnak, melynek keretében a megye vállalatai rendszeresen segítséget nyújtanak egy-egy iskolának. A nyári vakáció előtt új felhívás született: a számítógépes oktatás feltételeinek megteremtésére szólított fel. Hatására sok településen megmozdultak a termelő kollektívák. Orgoványon például a községi tanácsnál nyitottak e célra számlát, s olyan ütemben gyarapodott a pénz, hogy szeptemberben már nem egy, hanem két számítógépet vásárolhattak az általános iskolának. A kezdeményezésnek, a társadalmi összefogásnak köszönhetően ezen az őszön már az orgoványi gyerekek is megkezdhették a számítástechnika alapjainak elsajátítását.

Uránszennyeződés

Nagy-Britanniában a Harwell laboratórium kutatási programot kezdeményezett az integrált áramkörök katasztrofális meghibásodását előidéző urán és egyéb szennyezők eredetének kimutatására. Ezek radioaktivitása következtében keletkeznek ilyen hibák. A kibocsátott alfarészecskék által képviselt villamos töltések megváltoztathatják a memóriaalapáramkörök állapotát, vagy az aritmetikai áramkörök hibás működését idézhetik elő. Már többször előfordult, hogy a radioaktív szennyezőelemek következtében létrejövő számítógép-meghibásodás miatt egész üzemek működése automatikusan leállt.

A Harwell laboratóriumban kidolgozták azt az elemzési eljárást, amely minimális mennyiségű urán jelenlétét ki tudja mutatni. A vizsgálat során a félvezető anyagot poliamid fóliával vonják be és neutronsugárzás hatásának teszik ki. A hasadási termékeket fényérzékeny réteg regisztrálja, amelyet előhívás után mikroszkópon vizsgálva, pontosan meghatározható a jelenlevő urán mennyisége. Az eljárást felhasználják az integrált áramkörök gyártásához szükséges szilícium, gallium-arzenid és műanyag tokozások vizsgálatára.

Fuvardíj-elszámolás

1983—85 között a hazai intézmények tízezres nagyságrendben vásárolták a Commodore 64-eket professzionális célokra. Ismerve e gép korlátait, a FÜTI Mikroorg leányvállalata keresett egy olyan alkalmazást, amely nem igényel túl nagy adatforgalmat, a nyilvántartott tételszám százas nagyságrendű, s mégis gyakori, tipikus feladat. Így jutottak el a fuvardíj-elszámoló rendszerhez. Ez 120-150 gépkocsira elvégzi a tarifőri, statisztikai feladatokat, és a fuvarokmányokat is elkészíti. Ha a gépkocsiállomány nagyobb, akkor még egy hajlékonylemezes tárolót kell a géphez csatlakoztatni.

A rendszer gyorsan népszerűvé vált; alig néhány hónappal elkészülte után már tucatnyi helyen megvásárolták és alkalmazzák. Ez motiválta a kidolgozókat arra, hogy már IBM PC-vel kompatibilis gépre is átdolgozták.

Bővíthető hálózat

A végtelenségig bővíthető a SZÁMSZÖV Kisszövetkezet által az IBM PC/AT gépekre kifejlesztett, ötletes lokális hálózat. Két kategóriába tartozó AT-gépeket tartalmaz: van közöttük egyszerű munkahely és csoportvezérlő. Az utóbbihoz legfeljebb négy munkahely tartozhat, de lehet közöttük egy újabb csoportvezérlő is. A hálózaton belül bármely gépről bármely periféria elérhető.

LEGÚJABB
SZÁMÍTÁSTECHNIKAI KÍNÁLATUNK
NAGY VÁLASZTÉKÁVAL
VÁRJUK ÜGYFELEINKET
AZ ORGTECHNIK '86
KIÁLLÍTÁS
58-AS STANDJÁN

Fotoelektronik ISZ

Szki PRONET LOKÁLIS HÁLÓZAT

HARDVER

- A PROPER—16/A, PROPER—16/W, PROPER—16/m IBM PC kompatibilis számítógépcsaládból összeállítható hálózat kétirányú busz szervezésű
- A hálózat kiépítéséhez a PROPER—16 számítógépbe beduga-szolható lokális hálózati kártya szükséges
- A hálózat állomásai közös átviteli közegre kapcsolódnak
- Könnyű a hálózat konfigurálása és újabb állomások beiktatása

Főbb műszaki jellemzők

- adatátviteli sebesség: 1 Mbit/s
- adatátviteli közeg: csavart érpárú — vagy telefonkábel
- a főkábel hossza: maximum 500 m (ajánlott hossz max. 300 m)
- Az állomások elméleti maximális száma: 255 (a hálózatba kapcsolt erőforrásoktól és a felhasználás jellegétől függően ajánlott szám: 2-10)

SZOFTVER

- PROPOS V.3.02 vagy ezzel kompatibilis operációs rendszer
- Többfeladatos, többfelhasználós környezetben PROMOS operációs rendszer kiegészítés
- A hálózat szolgáltatásainak elérése külső parancsok segítségével, operátori konzolról vagy programból (felhasználói interfész) biztosított

AZ SZKI – STABIL PARTNER!

Számítástechnikai Kutató Intézet és Innovációs Központ

1251 Budapest, Pf. 19.

Információ: SCI—L Számítástechnikai Informatikai Fejlesztő Leányvállalat
1011 Budapest I., Iskola u. 10.
SCITEL Számítástechnikai Fejlesztő Leasing Leányvállalat
1015 Budapest I., Donáti u. 35—45.